

Universidad de las Ciencias Informáticas

FACULTAD 6



Trabajo de Diploma para optar por el título de Ingeniero en Ciencias
Informáticas

**Título: Sistema de Administración para el Sistema de Información
Geográfica de la Universidad de las Ciencias Informáticas**

Autor:

Rafael Calzado Maceo

Tutores:

MSc. Daniel Echevarría González

Ing. Yampier Medina Tarancón

“La Habana, 2015”

“Año 57 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año 2015.

Rafael Calzado Maceo

Firma del Autor

MSc. Daniel Echevarría González

Firma del Tutor

Ing. Yampier Medina Tarancón

Firma del Tutor

DATOS DE CONTACTO

El MSc. Daniel Echevarría González: graduado de Licenciatura en Geografía de la Facultad de Geografía de la Universidad de la Habana año 1995, Master en Ciencias en la Facultad de Geografía de la Universidad de la Habana año 1998, profesor asistente año 2012, actualmente se desempeña como especialista del Proyecto Aplicativos SIG.

El Ing. Yampier Medina Tarancón: ingeniero en Ciencias Informáticas graduado en el 2009 en la Universidad de las Ciencias Informáticas. Ha desempeñado los roles de planificador, asesor de calidad, jefe de grupo de calidad, probador y revisor. Diplomado en Docencia e Innovación Universitaria. Profesor Asistente. Impartió las asignaturas curriculares de Historia de la Informática, Gestión de Software, y Componente Profesional de Ingeniería y Gestión de Software. Además de las asignaturas optativas de Introducción a la Geoinformática e Infraestructura de Datos Espaciales.

AGRADECIMIENTOS

No me alcanzarían las palabras para agradecerles a todas las personas que me ayudaron y me apoyaron para que este día se convirtiera en una realidad. Si quisiera agradecerle a mi madre por estar ahí todos estos años y enseñarme, educarme y darme amor. A mi esposa por soportarme en mis momentos de estrés por quererme y por darme lo más importante que me ha sucedido en esta vida que es nuestra hija. A mi tía que ha sido como una madre para mí, siempre dando el sí ante cualquier situación que se me presentara. A mi hermano por estar ahí en todo momento y ayudarme a tomar decisiones difíciles en algunos momentos de mi vida. A mis suegros que me aceptaron en la familia como uno más de ellos sin diferencia alguna. Al grupo 6303 por estos hermosos años de convivencia y amistad. A mis abuelitos que hoy no están aquí físicamente pero sé que en cualquier parte que estén ahora están orgullosos de mí y que siempre me han dado fuerzas estén donde estén. A todas aquellas personas que hicieron posible este momento.

DEDICATORIA

Dedico este trabajo a aquellas personas que siempre me apoyaron en los momentos más difíciles de mi carrera y de la vida. A mi esposa, mi madre, mi hermano, mi tía, y mi primo, a aquellos que estuvieron ahí en todo momento.

RESUMEN

La presente investigación se basó en el desarrollo de un Sistema de Administración para el Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas, específicamente en el mejoramiento del proceso de gestión de la información que contiene dicho sistema. El Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas hoy no cuenta con una interfaz ni métodos de actualización para gestionar la información que en él se almacena, por ello que surge la necesidad de implementar un sistema que permita un mejor desempeño de las labores de gestión de la información con el fin de colocarlos a disposición de la comunidad universitaria. Esta investigación comprende un desarrollo progresivo de diferentes etapas que inicia desde la recopilación bibliográfica hasta las pruebas que se le hacen finalmente al software. Se emplearon una serie de técnicas e instrumentos para la recolección de datos, específicamente el análisis de fuentes documentales. Para la construcción del sistema y el cumplimiento de los objetivos planteados se empleó la metodología OpenUp. Adicionalmente, para el desarrollo de la aplicación se utilizaron diversas tecnologías como el lenguaje de programación PHP, el sistema manejador de base de datos PostgreSQL, el servidor Web Apache 2.2, el IDE de desarrollo NetBeans v8.0 para Linux, PgAdmin III es una aplicación gráfica de código abierto que brinda la posibilidad de administrar el Sistema Gestor de Base de Datos PostgreSQL, entre otras. De esta manera el proceso de gestión de la información se realizará de forma automatizada ahorrando esfuerzo y tiempo.

Palabras Claves: actualización, gestión, sistema de administración, sistema de información geográfica.

ABSTRACT

This research was based on the development of a Management System for the Geographic Information System of the University of Informatics Sciences, specifically on improving the information management process contained in this system. Today the Geographic Information System of the University of Informatics Sciences does not have an interface neither updating methods for managing information stored therein, which is why the need to implement a system that allows a better performance in terms of information management in order to place them at the disposal of the university community. This research is a progressive development of different stages that starts from the bibliography study up to the tests run to the software at the end. A number of techniques and tools for data collection, specifically the analysis of documentary sources were used. For the creation of the system and the fulfillment of the objectives the OpenUP methodology was used. Additionally, for the development of the application, various technologies such as PHP programming language, PostgreSQL database management system, Apache Web Server 2.2, the development IDE Netbeans v8.0 for Linux, PgAdmin III which is an open source graphical application that provides the ability to manage the PostgreSQL database management system and others were used. In this way, the information management process will be automated saving time and effort.

Keywords: update, management, management system, geographic information system.

ÍNDICE

DECLARACIÓN DE AUTORÍA	I
DATOS DE CONTACTO.....	II
AGRADECIMIENTOS	III
DEDICATORIA	IV
RESUMEN.....	V
ABSTRACT.....	VI
ÍNDICE	VII
ÍNDICE DE TABLAS	XI
ÍNDICE DE FIGURAS	XII
INTRODUCCIÓN.....	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN	5
1.1. Sistema de Información Geográfica	5
1.2. Sistemas existentes a nivel internacional.....	5
1.3. Sistemas existentes a nivel nacional.....	6
1.4. Metodología de desarrollo de software	7
1.4.1. OpenUp	8
1.5. Fundamentación de la metodología seleccionada	10
1.6. Tecnologías y herramientas para el desarrollo del sistema de administración para el SIGUCI	10
1.6.1. UML v2.0	10
1.6.2. Visual Paradigm for UML v8.0	11
1.7. Herramientas y lenguajes de programación.....	12
1.7.1. NetBeans v8.0 para Linux.....	12
1.7.2. PgAdmin III.....	13

1.8. Tecnologías web.....	13
1.8.1. Tecnologías del lado del cliente.....	14
1.8.2. Tecnologías del lado del servidor	15
1.9. Sistema gestor de base de datos.....	18
1.9.1. PostgreSQL v9.1	18
1.9.2. PostGIS v1.5	19
Conclusiones parciales	20
CAPÍTULO 2: ARQUITECTURA Y DISEÑO DE LA SOLUCIÓN PROPUESTA PARA EL SISTEMA DE ADMINISTRACIÓN PARA EL SIGUCI.....	21
2.1 Modelo de dominio	21
2.2.1. Descripción de las clases de dominio	22
2.2. Especificación de requisitos.....	23
2.2.1 Requisitos funcionales.....	23
2.2.2. Requisitos no funcionales	35
2.3. Modelo de casos de uso	37
2.3.1 Actores del Sistema.....	37
2.3.2. Diagrama de caso de uso del sistema	38
2.3.3. Descripción de los casos de uso.....	38
2.3.4. Descripción del Caso de Uso arquitectónicamente significativo del sistema	39
2.4. Modelo de diseño	43
2.4.1. Diagrama de paquetes.....	44
2.4.2. Diagramas de Clases del Diseño	45
2.4.3. Diagrama de Clases del Diseño del caso de uso Gestionar usuario.	45
2.5. Patrones utilizados para el desarrollo de la solución.....	46
2.5.1. Patrones de casos de uso	46

2.5.2. Patrones Arquitectónicos	46
2.5.3. Patrones de diseño	47
2.6. Modelo de Datos	49
2.6.1. Descripción de los principales objetos del modelo de datos	50
2.7. Modelo de despliegue	52
Conclusiones parciales	53
CAPÍTULO III: “IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA DE ADMINISTRACIÓN PARA EL SIGUCI”	54
3.1. Modelo de implementación	54
3.1.1. Diagrama de componentes	54
3.1.2. Código fuente	55
3.1.3. Estándar de codificación	55
3.1.4. Estilos de codificación utilizados	56
3.2. Pruebas del software	57
3.2.1. Niveles de prueba	57
3.2.2. Método de Prueba	57
3.2.3. Diseño de Casos de Prueba	58
3.3. Resultados de las pruebas	60
3.3.1. Caja Negra	60
3.3.2. Pruebas de Seguridad y Control de Acceso	61
3.3.3. Prueba de Usabilidad	62
Conclusiones parciales	63
Conclusiones generales	64
RECOMENDACIONES	65
REFERENCIAS BIBLIOGRÁFICAS	66

Glosario de Términos.....68

ÍNDICE DE TABLAS

Tabla 1 Descripción de las clases de dominio.....	22
Tabla 2 Descripción de los actores.	37
Tabla 3 Descripción del caso de uso Gestionar Usuario	39
Tabla 4 Descripción del objeto usuario.	50
Tabla 5 Descripción del objeto nodos.	51
Tabla 6. Descripción del objeto rol.	51
Tabla 7 Descripción de las variables del caso de uso "Gestionar Usuario".	58
Tabla 8 Escenario "Adicionar Usuario".....	59
Tabla 9 Descripción de las pruebas de seguridad y control de acceso	61

ÍNDICE DE FIGURAS

Figura. 1 Ciclo de vida de un proyecto según OpenUp	8
Figura. 2 Modelo de dominio.....	21
Figura. 3 Diagrama de caso de uso del sistema	38
Figura. 4 Diagrama de paquetes SIGADMIN	44
Figura. 5 Diagrama de clases del diseño del caso de uso Gestionar Usuario.	45
Figura. 6 Evidencia del patrón creador.....	47
Figura. 7 Modelo de Datos.....	50
Figura. 8 Modelo de despliegue	52
Figura. 9 Diagrama de componentes	55
Figura. 10 Ejemplo de empleo del estándar lowerCamelCase	56
Figura. 11 Resultados de las pruebas.....	61

INTRODUCCIÓN

Desde tiempos antiguos ya se era consciente de que algunos fenómenos se repiten en el espacio y en el tiempo. Teniendo en cuenta la necesidad de recordar las rutas relacionadas con la ubicación del agua y la caza, los primeros bocetos se crearon referidos al espacio.

La tecnología de los mapas ha tenido un progreso continuo con el fin de satisfacer las demandas de las nuevas generaciones de usuarios y de creadores de mapas. La información espacial se puede almacenar en base de datos espaciales, de donde se puede extraer. Estas herramientas permiten hacer mapas más dinámicos e interactivos, que pueden ser modificados digitalmente.

Un sistema de información geográfica (SIG), es un sistema para la gestión de conocimiento geográfico que se estructura en diferentes conjuntos de información. Permite la organización, almacenamiento, manipulación, análisis y modelación de grandes cantidades de datos procedentes del mundo real que están vinculados a una referencia espacial, facilitando la incorporación de aspectos sociales-culturales, económicos y ambientales que conducen a la toma de decisiones.

En el Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI), la Línea de Productos de Software (LPS) Aplicativos-SIG, es el grupo encargado de la construcción y personalización de Sistemas de Información Geográfica. Entre los principales productos desarrollados por la LPS, se encuentra el SIGUCI construido bajo un esquema diferente al desarrollo de otros sistemas similares que se implementan en la LPS, definido por la utilización de herramientas y tecnologías no aprovechadas anteriormente por el equipo de desarrollo.

Con la implementación del SIGUCI, la LPS obtuvo una aplicación parcialmente reutilizable para el desarrollo de otros SIG con características equivalentes a dicho sistema, a través de un diseño de software aplicado para obtener clases, módulos, complementos y una base de datos escalables desde la perspectiva de desarrollo y reutilización. Sin embargo, durante los últimos meses y debido a que la universidad ha estado llevando a cabo un proceso de mejoramiento de los servicios que se ofrecen así como la ejecución de otros totalmente nuevos como la instalación de puntos Wi-Fi; se ha generado nueva información que ha sido necesaria representar geográficamente en el SIGUCI, pero se ha visto afectado este proceso por algunas deficiencias que se mencionan a continuación:

- Debido a que el sistema no posee interfaces ni métodos de actualización y configuración de los componentes y variables que utiliza, hay que recurrir al equipo de la LPS para que agregue la nueva información que se va a mostrar, lo que conlleva mayor esfuerzo y tiempo.
- La falta de un control de acceso adecuado, imposibilita que los usuarios de las diferentes áreas (Residencia, Mantenimiento, Docencia, etc.) que representan información de interés para el resto de los usuarios que acceden al sistema, puedan realizar una administración y gestión de dicha información de forma independiente sin afectar la de las otras áreas.
- La publicación de información similar a otras disponibles en otros servicios (Directorio telefónico, Directorio de personas, Servicios telemáticos, Correo, etc.) conlleva a una duplicación de datos que impide una correcta interoperabilidad entre aplicaciones y servicios de la universidad.

Estas deficiencias afectan en gran medida, el objetivo principal de la LPS para con el SIGUCI, que es hacer más accesible aquella información que puede ser referenciada geográficamente y siendo utilizada esta por un SIG, permitir a los usuarios de las diferentes áreas que se agrupan en la universidad, acceder a una herramienta que apoye la toma de decisiones y que ofrezca un mecanismo viable de integración con la infraestructura informatizada que existe en la UCI.

Por lo expuesto anteriormente el **problema a resolver** queda definido en la siguiente interrogante:

¿Cómo mejorar el proceso de gestión de la información que se procesa en el SIGUCI?

El **objeto de estudio** está enmarcado en los procesos de administración de la información de los SIG, estableciendo como **campo de acción** el proceso de gestión de la información del SIGUCI.

Para dar solución al problema planteado se define como **objetivo general** de la investigación: Desarrollar un sistema de administración para el SIGUCI que permita la gestión de la información que se procesa en dicho sistema.

Para dar solución al **objetivo general** se definen los siguientes **objetivos específicos**:

- Construir el marco teórico sobre los mecanismos para la gestión de la información de los SIG.
- Realizar el análisis y diseño del sistema informático para la administración del SIGUCI.
- Realizar la implementación y prueba del sistema informático para la administración del SIGUCI.

Para el desarrollo de la investigación se tienen en cuenta las siguientes **preguntas científicas**:

- ¿Cómo se lleva a cabo actualmente el proceso de administración de la información de los Sistemas de Información Geográfica?
- ¿Cuáles son las dificultades que presenta el proceso de gestión de la información del SIGUCI?
- ¿Qué herramientas y tecnologías deben ser empleadas en el desarrollo del sistema de administración para el SIGUCI?
- ¿Cuáles son las características con las que debe contar el sistema de administración para el SIGUCI?
- ¿Cómo determinar que el sistema de administración para el SIGUCI cumple con las funcionalidades necesarias?

En función de dar cumplimiento a los objetivos planteados se definen las siguientes **tareas de investigación**:

- Caracterización del proceso de gestión de la información que procesan los Sistemas de Información Geográfica.
- Realización de un estudio de los sistemas existentes para la administración de la información de los Sistemas de Información Geográfica.
- Caracterización de las principales herramientas, tecnologías, lenguajes y metodologías a utilizar para la construcción de la solución propuesta.
- Realización del análisis y el diseño de la solución propuesta.
- Implementación y validación de la solución propuesta.

Métodos de investigación:

Métodos Teóricos:

Inductivo-Deductivo: A partir del problema identificado permitió plantear objetivos específicos y la idea a defender que es resuelta en el transcurso de la investigación mediante la aplicación de los métodos científicos.

Histórico-Lógico: Se emplea para llevar a cabo un estudio crítico sobre los modelos y metodologías de gestión de proyectos en cuanto a la realización de un análisis geo-referencial para el proceso de toma de decisiones.

Métodos Empíricos:

Observación: Este método se utiliza para conocer cómo se realiza el proceso de gestión de la información para el SIGUCI, así como para identificar los requisitos funcionales y no funcionales.

Análisis Documental: Se emplea para el estudio de la bibliografía especializada existente en el ámbito nacional e internacional permitiendo así la obtención de la información necesaria para la elaboración de la propuesta de solución.

Resultados esperados

- Sistema de Administración para el SIGUCI.
- Documentación técnica asociada a todo el proceso de desarrollo del software.

El presente trabajo se ha estructurado de la siguiente manera: Introducción, Capítulo I, Capítulo II, Capítulo III, Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos. Con el objetivo de lograr una mejor comprensión se presenta una pequeña descripción de los capítulos.

Capítulo I: “Fundamentación teórica de la investigación”

En este capítulo se presentan los principales conceptos a investigar asociados al dominio del problema. Se realiza un estudio del estado del arte sobre las soluciones existentes a nivel nacional e internacional. Además se fundamentan las tecnologías, herramientas, así como los lenguajes de programación que se utilizan para el desarrollo de la solución propuesta.

Capítulo II: “Arquitectura y Diseño del Sistema de Administración para el Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas”

En este capítulo se realiza una descripción del negocio teniendo en cuenta los requisitos funcionales y no funcionales, así como los artefactos generados para el desarrollo del sistema.

Capítulo III: “Implementación y Prueba del Sistema de Administración para el Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas”

En este capítulo como resultado del diseño antes realizado se muestra el modelo de implementación, así como el diagrama de componentes de la solución propuesta. Con el objetivo de comprobar que el sistema funciona de la mejor manera se realizan pruebas las cuales se describen en este capítulo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

Este capítulo se realizará un estudio de los elementos teóricos que sirven de base para el cumplimiento del objetivo propuesto en la presente investigación. Se definen los conceptos asociados al dominio del problema a partir del criterio asumido por diferentes autores. Se realiza un estudio de las características de los sistemas de administración de SIG existentes así como sus aplicaciones en los procesos de toma de decisiones. Además se fundamentan las tecnologías, herramientas, así como los lenguajes de programación que se utilizan para el desarrollo de la solución propuesta.

1.1. Sistema de Información Geográfica

Un sistema de información geográfica (también conocido con los acrónimos SIG en español o *GIS* en inglés) es un conjunto de herramientas que integra y relaciona diversos componentes (usuarios, hardware, software, procesos) que permiten la organización, almacenamiento, manipulación, análisis y modelización de grandes cantidades de datos procedentes del mundo real que están vinculados a una referencia espacial, facilitando la incorporación de aspectos sociales-culturales, económicos y ambientales que conducen a la toma de decisiones de una manera más eficaz (1).

A consideración del autor, un SIG es cualquier sistema de información capaz de almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada, con el fin de apoyar la toma de decisiones en un área específica.

1.2. Sistemas existentes a nivel internacional

A nivel internacional existen varios sistemas que garantizan la administración de los SIG, entre los que se encuentran:

Esri (Environmental Systems Research Institute por sus siglas en inglés) es el proveedor líder en Sistemas de Información Geográfica (SIG) siendo ArcGIS la tecnología más implantada dentro del ámbito de la Administración Pública española. Con su tecnología proporcionan los medios necesarios para que diferentes organismos puedan abordar sus proyectos con componente geográfica, maximizando el uso de ésta en los procesos de toma de decisiones (2).

La tecnología SIG de Esri proporciona capacidades para administrar, analizar y servir grandes volúmenes de información. Los principales usuarios dentro del marco público se enmarcan en las áreas de medio ambiente, agricultura, análisis estadísticos, o dentro del ámbito puramente catastral o urbanístico. Aunque, cada vez más emergen nuevos usos como la gestión sanitaria, de infraestructuras, o de seguridad y emergencias. Además permite interconexión entre administraciones públicas, basada en modelos de negocio interadministrativos, como son las bases de datos federadas.

Esta tecnología permite a nuestros usuarios poder interconectar diferentes fuentes de datos, inclusive de diferentes fabricantes (motores de datos) y de otras administraciones, como una fuente única, evitando la gestión de datos innecesarios o duplicidades e inconsistencias. Esri, está comprometida con la distribución y colaboración con las Administraciones públicas en su inercia hacia la puesta en marcha de sistemas interoperables que hagan accesible toda la información precisa no sólo al resto de instituciones Españolas, sino Europeas.

ArcGIS for Desktop es el principal producto que emplean los profesionales de SIG para compilar, utilizar y administrar información geográfica.

ArcGIS for Desktop incluye completas aplicaciones SIG profesionales que facilitan diversas tareas de SIG como, por ejemplo, representación cartográfica, compilación de datos, análisis, administración de geodatos e imágenes, y uso compartido de información geográfica. (3)

Además es la plataforma que los profesionales de SIG utilizan para administrar los proyectos y flujos de trabajo de SIG, así como para crear datos, mapas, modelos y aplicaciones. Además, constituye el punto de partida y la base para la implementación de SIG en las organizaciones y en la Web. ArcGIS admite bases de datos multiusuario donde varios usuarios utilizan y editan los datos de forma simultánea. Esto permite el acceso, la administración y la actualización de la información por parte de usuarios distintos en numerosos grupos de trabajo y departamentos. Por ejemplo, usuarios administrativos y trabajadores de campo pueden actualizar la información al mismo tiempo, y cada grupo verá casi de inmediato los cambios introducidos por sus colegas.

1.3. Sistemas existentes a nivel nacional

En Cuba se trabaja en la creación de la Infraestructura de Datos Espaciales de la República de Cuba (IDERC) cuya visión es compartir la información geográfica en un ambiente cooperativo interinstitucional

en función de la toma de decisiones económicas, políticas, ambientales, etc. El Portal Geo-espacial Nacional producto de esta iniciativa fue lanzado con carácter experimental en Noviembre de 2004. (4)

Además está el caso del Sistema de Información Geográfica para la gestión de la estadística de salud de Cuba (SIG-ESAC), con el objetivo de facilitar la gestión de la estadística de salud el cual permite cartografiar y hacer diferentes tipos de análisis de importantes indicadores de salud, morbilidad, demográficos, recursos y servicio.

En la UCI, en el departamento de GEYSED están conscientes de la problemática de la administración del SIG_UCI donde la principal dificultad radica no sólo en los escasos medios con los que en muchos casos cuentan, sino que el sistema no posee interfaces ni métodos de actualización y configuración de los componentes y variables que utiliza. Para lo cual, se propone como solución la creación de un Sistema de Administración para el Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas basada en la idea de que permita la gestión de la información que se procesa en dicho sistema.

1.4. Metodología de desarrollo de software

En la actualidad no es posible desarrollar una aplicación informática eficiente sin que el equipo de trabajo esté orientado por una metodología de desarrollo de software, estas imponen un proceso disciplinado que tiene como principal objetivo aumentar la calidad del software que se produce en cada una de sus fases de desarrollo. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque diferente para las actividades que tienen lugar durante el proceso (5).

Por una parte se tiene aquellas propuestas más tradicionales que se centran especialmente en el control del proceso estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán. Dentro de esta clasificación se encuentra Rational Unified Process (RUP). Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas y una de las más utilizadas es OpenUP.

1.4.1. OpenUp

El OpenUp es un proceso modelo y extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo (6)

Permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito. El ciclo de vida de un proyecto, según la metodología OpenUP permite que los integrantes del equipo de desarrollo aporten micro-incrementos, que pueden ser el resultado del trabajo de unas pocas horas o unos pocos días. El progreso se puede visualizar diariamente, ya que la aplicación va evolucionando en función de estos pequeños aportes.

El objetivo de OpenUP es ayudar al equipo de desarrollo a lo largo de todo el ciclo de vida de las iteraciones para que sea capaz de añadir valor de negocio a los clientes de una forma predecible con la entrega de un software operativo y funcional al final de cada iteración. El ciclo de vida del proyecto provee a los clientes de una visión del proyecto, transparencia y los medios para que controlen la financiación, el riesgo, el ámbito, el valor de retorno esperado, etc. (7).

Todo proyecto en OpenUP consta de cuatro fases: inicio, elaboración, construcción y transición. Cada una de estas fases se divide a su vez en iteraciones. A continuación se muestran estas fases aplicadas al proceso de construcción del sistema de administración para el SIGUCI.

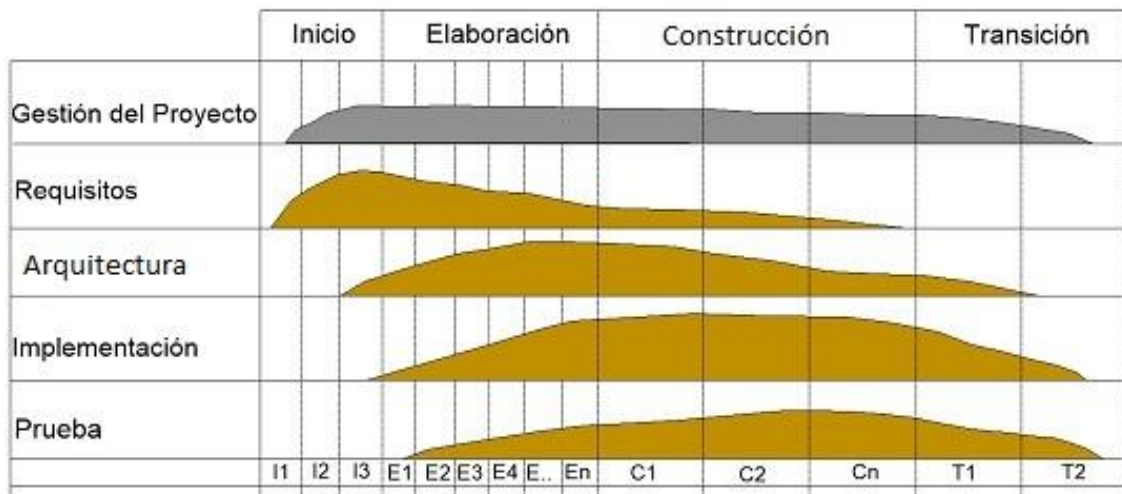


Figura. 1 Ciclo de vida de un proyecto según OpenUp

Fase de inicio: En esta fase se tomaron en cuenta y quedaron plasmados en los objetivos del proyecto las necesidades de cada participante del proyecto, quedando definidos los límites, criterios de aceptación, casos de uso críticos y una estimación inicial del tiempo requerido para el desarrollo del sistema.

Fase de elaboración: El análisis del dominio y la definición de la arquitectura del sistema son algunas de las tareas que se realizan en esta fase. Además de elaborar un plan de proyecto estableciendo los requisitos y una arquitectura estable. Por otra parte también se especifican las herramientas, el entorno de desarrollo y la infraestructura a utilizar para el proceso de desarrollo del sistema de administración para el SIGUCI. Al final de la fase se asume una definición clara y precisa de los casos de uso, la arquitectura del sistema y el prototipo ejecutable de la misma.

Fase de construcción: Fueron realizados, probados e integrados a esta fase todos los componentes y funcionalidades que faltaban por implementar. Los resultados obtenidos en forma de incrementos ejecutables fueron desarrollados de la forma más rápida posible sin dejar a un lado la calidad que requiere el desarrollo del sistema.

Fase de transición: Esta fase corresponde a la entrega del producto a los usuarios finales que serán los encargados de utilizar y dar mantenimiento al sistema a los que se les dio una capacitación sobre el funcionamiento del sistema, además de realizarles pequeñas pruebas al software. Una vez demostrado que el sistema estaba lo suficientemente maduro se procedió a la entrega final.

Principios

- Colaborar para sincronizar intereses y compartir conocimiento. Este principio promueve prácticas que impulsan un ambiente de equipo saludable, facilitan la colaboración y desarrollan un conocimiento compartido del proyecto.
- Equilibrar las prioridades para maximizar el beneficio obtenido por los interesados en el proyecto. Este principio promueve prácticas que permiten a los participantes de los proyectos desarrollar una solución que maximice los beneficios obtenidos por los participantes y que cumple con los requisitos y restricciones del proyecto.
- Centrarse en la arquitectura de forma temprana para minimizar el riesgo y organizar el desarrollo.
- Desarrollo evolutivo para obtener retroalimentación y mejoramiento continuo. Este principio promueve prácticas que permiten a los equipos de desarrollo obtener retroalimentación temprana y

continua de los participantes del proyecto, permitiendo demostrarles incrementos progresivos en la funcionalidad.

1.5. Fundamentación de la metodología seleccionada

Como metodología de desarrollo de software para el desarrollo del sistema de administración para el SIGUCI se ha seleccionado OpenUp por ser un proceso completo, flexible y corto. Para atender las necesidades que no están cubiertas en su contenido OpenUp es extensible a ser utilizado como base sobre la cual se pueden añadir o adaptarse a contenido de otro proceso que sea necesario, además que cubre aspectos como la seguridad y contratación de personal, fomenta el uso de técnicas ágiles y principios mientras que tiene un ciclo de vida estructurado y probado que hace referencia en la continua entrega de software de calidad.

En muchos métodos de desarrollo ágil está contemplado en *feedback* del cliente como parte estructural del proceso de desarrollo, lo que favorece que el producto final se ajuste más a lo que el cliente necesita y no a lo que el cliente pidió en un primer momento.

1.6. Tecnologías y herramientas para el desarrollo del sistema de administración para el SIGUCI

1.6.1. UML v2.0

Es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar los artefactos del sistema de administración para el SIGUCI. Además de entender, diseñar, configurar, mantener y controlar la información sobre el sistema en construcción.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes (8).

Principales características

- Es un lenguaje consolidado, fácil de aprender y permite la documentación de todo el ciclo de desarrollo del sistema.
- Puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser usada para comunicarse con los usuarios o entre el equipo de desarrollo.
- Se ha venido adoptando en diferentes medios empresariales y académicos como el lenguaje estándar para el análisis y diseño de los sistemas de software.

La utilización del UML en la construcción del sistema de administración para el SIGUCI, permitió generar los diagramas y la documentación de todo el ciclo de desarrollo del software con vista a facilitar el desarrollo de futuras versiones o mejoras, optimizando el tiempo total de desarrollo. Contribuyendo además al aumento de la calidad y un mejor soporte al planeamiento y control del sistema.

1.6.2. *Visual Paradigm for UML v8.0*

Es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. *Visual Paradigm for UML* ha sido concebido para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas UML. Constituye una herramienta con doble licencia disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: *Enterprise, Professional, Community, Standard, Modeler y Personal* (9).

Se escoge *Visual Paradigm for UML* como herramienta Case para el modelado de diagramas de todo el ciclo de vida del software por las siguientes características:

- Disponibilidad en múltiples plataformas (*Windows, Linux*).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común por todo el equipo de desarrollo facilitando la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, con diferentes especificaciones.

- Licencia: gratuita.
- Soporta aplicaciones Web.

1.7. Herramientas y lenguajes de programación

1.7.1. NetBeans v8.0 para Linux

NetBeans IDE es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento. *Sun Microsystems* fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos (10).

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para varios lenguajes de programación. También está disponible *NetBeans Platform*; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Ventajas:

- La plataforma NetBeans puede ser usada para desarrollar cualquier tipo de aplicación.
- Reutilización del Módulos.
- Instalación y actualización simple.
- Incluye Templates y Wizards.
- Posee soporte para php.

Características principales de la plataforma:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.
- Framework basado en asistentes (diálogo paso a paso).

Con el uso de IDE de desarrollo NetBeans se facilitó el trabajo de la programación para la construcción del sistema de administración para el SIGUCI. Este IDE de desarrollo aunque fue creado inicialmente para Java, hoy tiene soporte para disímiles lenguajes de programación entre los que se encuentra PHP, lenguaje utilizado en la construcción del sistema. Además de que permite la instalación de pluguins y la utilización de librerías lo cual facilitó aún más el trabajo. Para la construcción del sistema fue utilizado un pluguins para facilitar la ejecución de los comandos del framework Symfony mediante una interfaz ahorrando el trabajo de tener que trabajar con la consola y así contribuir a un mejor desarrollo del sistema.

1.7.2. PgAdmin III

PgAdmin III es una aplicación gráfica de código abierto que ofrece la posibilidad de administrar el SGBD PostgreSQL. Posee herramientas de consultas SQL, editor de código procedural y agente de planificación SQL y puede ser ejecutado bajo diversas plataformas. No solo está diseñada para responder a consultas SQL simples, puede además desarrollar bases de datos complejas (11).

PgAdmin está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL sencillas para el desarrollo de bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de resaltado de sintaxis SQL, un editor de código del lado del servidor, un agente de planificación de tareas, el apoyo a la SlonyI motor de replicación y mucho más. La conexión al servidor se puede hacer a través de TCP / IP o Unix Domain hembra (en plataformas * nix), y puede ser encriptada SSL para la seguridad. No se requieren controladores adicionales para comunicarse con el servidor de base de datos.

1.8. Tecnologías web

Las tecnologías web poseen una significación preponderante por el papel que está jugando el internet en el mundo moderno. Esta plataforma ha ido evolucionando paulatinamente para convertirse en un ambiente donde se implementan potentes aplicaciones cliente/servidor o arquitecturas de n capas, unido a esto han ido surgiendo nuevas tecnologías que se relacionan con el desarrollo web lo que hacen a éste más interactivo e interesante. Entre las tecnologías utilizadas para la creación y mantenimientos de sitios web, están las que funcionan del lado del cliente y las del lado del servidor.

1.8.1. Tecnologías del lado del cliente

HTML

HTML, siglas de *Hyper Text Markup Language* («lenguaje de marcas de hipertexto»), es un lenguaje de etiquetas (también llamado lenguaje de marcado). Las páginas web habituales están formadas por cientos o miles de pares de etiquetas. De hecho, las letras "ML" de la sigla HTML significan "*markup language*", que es como se denominan en inglés a los lenguajes de marcado. Además de HTML, existen muchos otros lenguajes de etiquetas como XML, SGML, DocBook y MathML.

La principal ventaja de los lenguajes de etiquetas es que son muy sencillos de leer y escribir por parte de las personas y de los sistemas electrónicos. La principal desventaja es que pueden aumentar mucho el tamaño del documento, por lo que en general se utilizan etiquetas con nombres muy cortos (12).

JavaScript

Es un lenguaje interpretado, no requiere compilación. Fue creado por Netscape para incrementar las funcionalidades del lenguaje HTML. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en la página HTML y ejecutarlas adecuadamente. El código JavaScript es embebido directamente en el código HTML con las tags `<script>` y `</script>`, haciendo fácil la creación de páginas web con contenido dinámico. Los navegadores son los encargados de interpretar el código, por lo que no se requiere tener instalado ningún *framework*.

Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, entre otros y para la interacción con páginas web se provee de una implementación del Modelo de Objetos de Documento (Document Object Model, por sus siglas en inglés). En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Eguiluz, 2009).

JavaScript es un lenguaje sencillo y fácil de aprender, el cual puede hacerse cargo de gran cantidad de funcionalidades del lado del cliente, liberando al servidor de esas responsabilidades. Es rápido y potente, ideal para implementar pequeñas funciones dentro de la página web.

En la construcción del sistema de administración para el SIGUCI se hace uso de JavaScript en las llamadas dinámicas, personalización y efectos especiales de la interfaz de usuario, en las validaciones de datos que un usuario entra por el formulario antes de enviarlos, además incluye los elementos necesarios

para que los Scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

CSS (Hojas de estilo en cascada)

Hojas de Estilo en Cascada (*Cascading Style Sheets*), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos (13).

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

JQuery v1.9.1

JQuery es un framework JavaScript que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Implementa una serie de clases que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, ya que funcionan de exacta forma en todas las plataformas más habituales, así pues, este framework JavaScript, nos ofrece una infraestructura con la que tendremos mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Además de esta y otras ventajas que posee, el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. Para ello simplemente tendremos que incluir en nuestras páginas un script JavaScript que contiene el código de JQuery, que podemos descargar de la propia página web del producto y comenzar a utilizar el *framework* (19).

1.8.2. Tecnologías del lado del servidor

Symfony v2.3.7

Es un framework creado completamente en PHP5 y diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Separa la lógica del negocio, la lógica del servidor y la presentación web. Con este framework el desarrollador podrá olvidarse de las

tareas comunes en la web y centrarse solo en las específicas del proyecto en que esté trabajando. Entre sus características más destacadas están: (14)

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de base de datos.
- Utiliza programación orientada a objetos.
- Utiliza un modelo MVC (Modelo vista controlador).
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Está preparado para aplicaciones empresariales.

Para el desarrollo del sistema de administración para SIGUCI se utilizó este framework ya que permite la creación de sitios web rápidos y seguros de forma profesional. Ya que es un proyecto de software libre la comunidad ayuda al desarrollo y depuración del framework.

Debido a la comunidad tan activa que posee esta herramienta, existe un sinfín de bundles o pequeños programas que proveen utilidades y funciones específicas para expandir la funcionalidad de nuestras aplicaciones, utilidades que son fáciles de instalar e implementar. Además de que este framework ofrece la facilidad de generar el *CRUD* de todas y cada una de las entidades que se tenga en un proyecto, lo cual reduce notablemente el tiempo de desarrollo.

PHP v5.3

Es un acrónimo recurrente que significa Procesador de Hipertexto (*Hypertext Pre-processor*), es un lenguaje de secuencia de comandos de servidor diseñado específicamente para la web. Es un producto de código abierto, es decir, que permite acceder a su código, utilizarlo, modificarlo y redistribuirlo (PHP, 2013).

Además este lenguaje es ejecutado en el servidor lo que permite acceder a los recursos que tenga éste, del mismo modo sólo el resultado de su ejecución es enviado al navegador y es normalmente una página HTML. En esto radica la ventaja de que no sea necesario que PHP sea soportado por el navegador al ser independiente de éste. El objetivo fundamental del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas, para esto se pueden realizar consultas a bases de datos, crear imágenes, leer y escribir en archivos y comunicarse con servidores remotos.

Algunas de las características más relevantes de este lenguaje son su rapidez a pesar de ser interpretado, su soporte multiplataforma tanto de diversos sistemas operativos, como servidores de aplicaciones y bases de datos (15).

¿Por qué se eligió PHP como lenguaje de programación para implementar la solución propuesta?

- Está soportado en la mayoría de las plataformas de Sistemas Operativos.
- Soporta una gran cantidad de Bases de Datos.
- Brinda todas las prestaciones necesarias y requeridas para el desarrollo del sistema propuesto.
- El PHP no tiene costo oculto, cuenta con un grupo de bibliotecas importantes y en caso de necesitar alguna es posible encontrarla de forma rápida y gratis en Internet.

Apache v2.2

El servidor HTTP Apache es un servidor web de código abierto, para plataformas Unix (BSD, GNU/Linux, etc), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual (16).

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. Es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

¿Por qué se eligió Apache como servidor web para implementar la solución propuesta?

- Modular
- Código abierto
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

1.9. Sistema gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: DataBase Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejar los datos.

Los SGBD relacionales son una herramienta efectiva que permite a varios usuarios acceder a los datos al mismo tiempo. Brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos.

1.9.1. PostgreSQL v9.1

PostgreSQL es un SGBD objeto-relacional, orientado a objetos y multiplataforma dirigido por una comunidad de desarrolladores y organizaciones comerciales denominada PGDG (PostgreSQL Global Development Group). Se puede considerar el gestor de base de datos de código abierto más avanzado que se puede utilizar hoy en día.

Además ofrece soporte para el lenguaje SQL92\SQL3, integridad de transacciones y extensibilidad de tipos de datos, sirve de soporte a los lenguajes de programación más populares como PHP, C, C++, Java, Python, entre otros. El número de base de datos que puede contener es ilimitada y posee gran escalabilidad y rendimiento bajo grandes cargas de trabajo.

PostgreSQL puede aportar potencia y flexibilidad con características como las restricciones (*constraints*), disparador es (*triggers*) y reglas (*rules*), del mismo modo al incorporar los conceptos de clase, herencia, tipos y funciones se alcanza extender fácilmente el sistema. Dentro de los módulos con los que cuenta se encuentra PostGIS, de gran utilidad a la hora de trabajar con bases de datos espaciales. Esta tipología de base de datos se diferencia por contener además información referente a la localización espacial de objetos geográficos, elementos que tienen forma y ubicación en el espacio (17).

1.9.2. PostGIS v1.5

PostGIS es el módulo espacial de PostgreSQL que permite la manipulación y almacenamiento de datos espaciales. Además es capaz de implementar metadatos, funciones geométricas y topológicas para el trabajo con datos espaciales basados es el estándar del OpenGis Consortium. PostGIS es capaz de tratar grandes volúmenes de datos con escalabilidad y puede usarse con adaptaciones, en cualquier plataforma. Es de código abierto, distribuido bajo la licencia GNU (*General Public License*).

Dentro de las características que lo hacen ser el sistema idóneo para trabajar con SIG se puede mencionar que utiliza las librerías Proj4 para dar soporte para reproyección dinámica de coordenadas. Asegura además la integridad de los datos y maximiza el rendimiento al realizar una representación reducida de la geometría (18).

Fundamentación del gestor seleccionado

PostgreSQL con PostGIS ofrece la mejor solución para un desarrollo con soporte para datos y funciones SIG de gran interoperabilidad libre y de alto rendimiento, ofrecer garantía de integridad de los datos. Además incorpora funciones de diversa índole tales como el manejo de fechas, datos geométricos, etc. Otras de las características que conllevaron a la elección del gestor PostgreSQL son:

- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Permite la herencia entre tablas
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Conclusiones parciales

El estudio de los conceptos fundamentales y documentación referente a los sistemas de administración para SIG fue el punto de partida para una mejor comprensión de la solución que se desea desarrollar. El estudio de las diferentes metodologías de software permitió seleccionar a OpenUp como metodología de desarrollo, Visual Paradigm for UML como herramienta para el modelado de los artefactos generados empleando el lenguaje de modelado UML 2.0. Para la implementación de la solución se hará uso de NetBeans IDE 8.0, usando como lenguajes de programación PHP 5.3 soportado por los marcos de trabajo de Symfony 2.3.7 y JavaScript soportado por el marco de trabajo JQuery para el desarrollo del cliente de la aplicación. Para el manejo de la base de datos se utilizará PostgreSQL 9.1 y PostGIS 1.5 como módulo espacial de PostgreSQL que permite la manipulación y almacenamiento de datos espaciales y PgAdmin III como administrador del SGBD PostgreSQL. Además de Apache 2.2 como servidor web.

CAPÍTULO 2: ARQUITECTURA Y DISEÑO DE LA SOLUCIÓN PROPUESTA PARA EL SISTEMA DE ADMINISTRACIÓN PARA EL SIGUCI

En el presente capítulo se identifican las clases de dominio y las relaciones que existen entre ellas. Se realiza un análisis de las características con que debe contar el sistema identificando los requisitos funcionales y no funcionales. Se realiza el modelado de los casos de uso del sistema así como su descripción textual. Se analizarán los estilos arquitectónicos y los patrones de diseño. Mediante el diagrama de despliegue se especifica la estructura física de la solución propuesta.

2.1 Modelo de dominio

Modelo de dominio o conceptual: Se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, puede ser tomado como el punto de partida para el diseño del sistema. Cuando se realiza la programación orientada a objetos, el funcionamiento interno del software va a imitar en alguna medida a la realidad. (20).

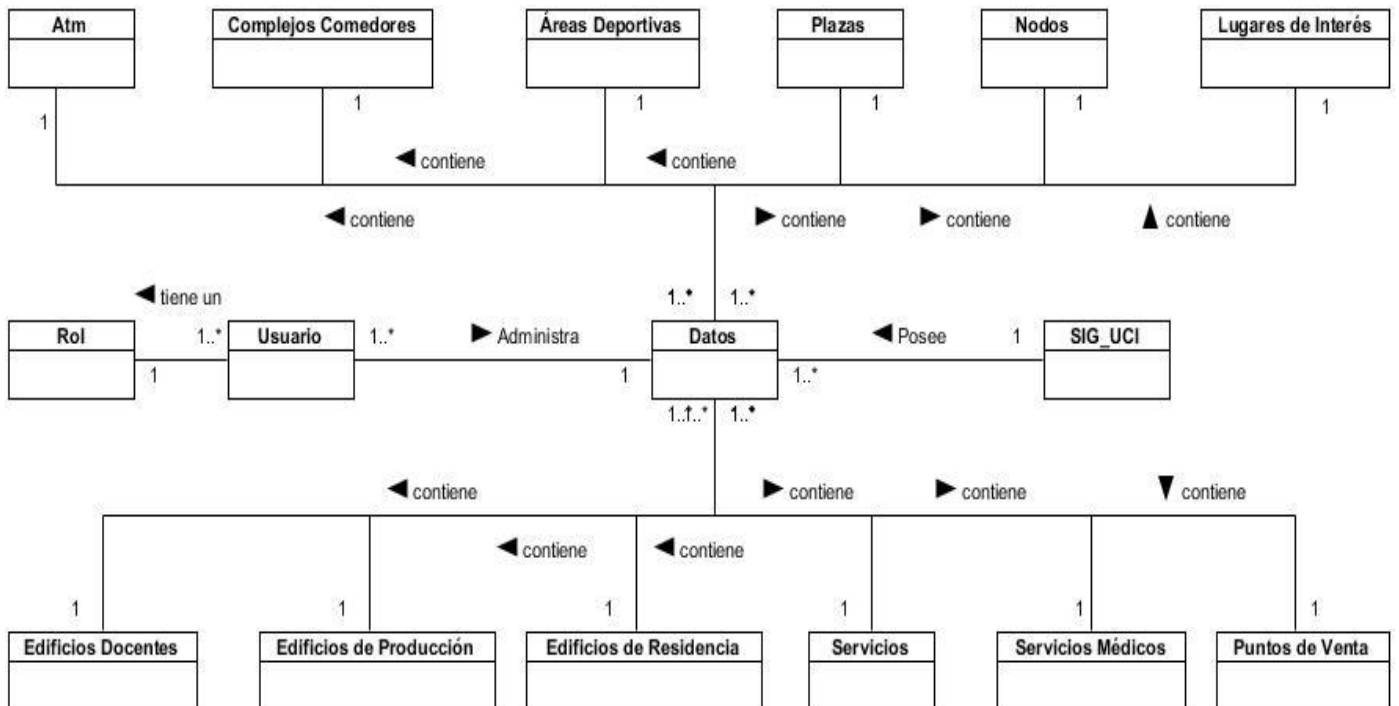


Figura. 2 Modelo de dominio

2.2.1. Descripción de las clases de dominio

Tabla 1 Descripción de las clases de dominio.

Clase	Descripción
Usuario	Persona que interactúa con el sistema y administra los datos existentes en la base de datos del SIGUCI.
Datos	Información almacenada en la base de datos del SIGUCI.
SIGUCI	Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas.
Rol	Papel desempeñado por los usuarios lo cual permite, trabajar con el sistema en dependencia de los privilegios asignados.
Atm	Entidad que almacena los datos de una cajero automático
Complejos Comedores	Entidad que almacena los datos de un complejo comedor
Áreas Deportivas	Entidad que almacena los datos de un área deportiva
Plazas	Entidad que almacena los datos de una plaza
Nodos	Entidad que almacena los datos de un nodo
Lugares de Interés	Entidad que almacena los datos de un lugar de interés
Edificios Docentes	Entidad que almacena los datos de un edificio docente
Edificios de Producción	Entidad que almacena los datos de un edificio de producción
Edificios de Residencia	Entidad que almacena los datos de un edificio de residencia
Servicios	Entidad que almacena los datos de un servicio
Servicios Médicos	Entidad que almacena los datos de un servicio médico
Puntos de Venta	Entidad que almacena los datos de un punto de venta

2.2. Especificación de requisitos

Los requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento (21).

2.2.1 Requisitos funcionales

Expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento). (21)

RF1. Autenticar Usuario: El usuario debe identificarse para poder acceder al sistema.

RF2. Adicionar Usuario

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo usuario.

Entrada: Datos del usuario que se va a adicionar (nombre, usuario, contraseña, rol).

Salida: El usuario queda adicionado en el sistema y a partir de ese momento podrá ejecutar acciones en el sistema según su rol.

RF3. Modificar Usuario

Descripción: El sistema a través de una interfaz debe permitir modificar los datos del usuario en cuestión.

Entrada: Datos del usuario que se va a modificar (nombre, usuario, contraseña, rol).

Salida: No procede.

RF4. Listar Usuario

Descripción: El sistema a través de una interfaz debe permitir visualizar los usuarios que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los usuarios registrados en el sistema.

RF5. Eliminar Usuario

Descripción: El sistema a través de una interfaz debe permitir visualizar los usuarios que están registrados en el sistema para seleccionar uno y eliminarlo.

Entrada: Identificador del usuario seleccionado.

Salida: Muestra el listado de los usuarios registrados en el sistema actualizado.

RF6. Buscar Usuario

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los usuarios que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del usuario seleccionado.

Salida: Muestra los datos del usuario seleccionado.

RF7. Adicionar Atm (Cajero Automático)

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo atm.

Entrada: Datos del atm que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El atm queda adicionado en el sistema.

RF8. Modificar Atm

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del atm en cuestión.

Entrada: Datos del atm que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el atm ya actualizado.

RF9. Listar Atm

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los atm que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los atm registrados en el sistema.

RF10. Eliminar Atm

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los atm que están registrados en el sistema para seleccionar uno y eliminarlo.

Entrada: Identificador del atm seleccionado.

Salida: Muestra el listado de los atm registrados en el sistema actualizado.

RF11. Buscar Atm

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los atm que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del atm seleccionado.

Salida: Muestra los datos del atm seleccionado.

RF12. Adicionar Nodos

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo nodo.

Entrada: Datos del nodo que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El nodo queda adicionado en el sistema.

RF13. Modificar Nodos

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del nodo en cuestión.

Entrada: Datos del nodo que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el nodo ya actualizado.

RF14. Listar Nodo

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los nodos que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los nodos registrados en el sistema.

RF15. Eliminar Nodo

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los nodos que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del nodo seleccionado.

Salida: Muestra el listado de los nodos registrados en el sistema actualizado.

RF16. Buscar Nodo

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los nodos que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del nodo seleccionado.

Salida: Muestra los datos del nodo seleccionado.

RF17. Adicionar Plaza

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear una nueva plaza.

Entrada: Datos de la plaza que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: La plaza queda adicionada en el sistema.

RF18. Modificar Plaza

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos de la plaza en cuestión.

Entrada: Datos de la plaza que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con la plaza actualizada.

RF19. Listar Plazas

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar las plazas que están registradas en el sistema.

Entrada: No procede.

Salida: Muestra el listado de las plazas registradas en el sistema.

RF20. Eliminar Plaza

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar las plazas que están registradas en el sistema, para seleccionar una y eliminarla.

Entrada: Identificador de la plaza seleccionada.

Salida: Muestra el listado de las plazas registradas en el sistema actualizada.

RF21. Buscar Plaza

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar las plazas que están registrados en el sistema, una vez seleccionada una, debe permitir mediante una interfaz visualizar los datos de esta.

Entrada: Identificador de la plaza seleccionada.

Salida: Muestra los datos de la plaza seleccionada.

RF22. Adicionar Complejos Comedores

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo complejo comedor.

Entrada: Datos del complejo que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El complejo comedor queda adicionado en el sistema.

RF23. Modificar Complejos Comedores

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del complejo en cuestión.

Entrada: Datos del complejo que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el complejo comedor actualizado.

RF24. Listar Complejos Comedores

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los complejos comedores que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los complejos comedores registrados en el sistema.

RF25. Eliminar Complejos Comedores

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los complejos comedores que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del complejo seleccionado.

Salida: Muestra el listado de los complejos comedores registrados en el sistema actualizado.

RF26. Buscar Complejos Comedores

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los complejos que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del complejo comedor seleccionado.

Salida: Muestra los datos del complejo comedor seleccionado.

RF27. Adicionar Área Deportiva

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear una nueva área deportiva.

Entrada: Datos del área deportiva que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El área deportiva queda adicionada en el sistema.

RF28. Modificar Área Deportiva

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del área deportiva en cuestión.

Entrada: Datos del área deportiva que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el área deportiva ya actualizada.

RF29. Listar Área Deportiva

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar las áreas deportivas que están registradas en el sistema.

Entrada: No procede.

Salida: Muestra el listado de las áreas deportivas registradas en el sistema.

RF30. Eliminar Área Deportiva

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar las áreas deportivas que están registradas en el sistema, para seleccionar una y eliminarla.

Entrada: Identificador del área deportiva seleccionada.

Salida: Muestra el listado de las áreas deportivas registradas en el sistema actualizada.

RF31. Buscar Área Deportiva

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar las áreas deportivas que están registradas en el sistema, una vez seleccionada una, debe permitir mediante una interfaz visualizar los datos de esta.

Entrada: Identificador del área deportiva seleccionada.

Salida: Muestra los datos del área deportiva seleccionada.

RF32. Adicionar Edificio Docente

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo edificio docente.

Entrada: Datos del edificio docente que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El edificio docente queda adicionado en el sistema.

RF33. Modificar Edificio Docente

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del edificio docente en cuestión.

Entrada: Datos del edificio docente que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el edificio docente actualizado.

RF34. Listar Edificio Docente

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios docentes que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los edificios docentes registrados en el sistema.

RF35. Eliminar Edificio Docente

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios docentes que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del edificio docente seleccionado.

Salida: Muestra el listado de los edificios docentes registrados en el sistema actualizado.

RF36. Buscar Edificio Docente

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios docentes que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del edificio docente seleccionado.

Salida: Muestra los datos del edificio docente seleccionado.

RF37. Adicionar Edificio Producción

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo edificio de producción.

Entrada: Datos del edificio de producción que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El edificio de producción queda adicionado en el sistema.

RF38. Modificar Edificio Producción

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del edificio de producción en cuestión.

Entrada: Datos del edificio de producción que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el edificio de producción actualizado.

RF39. Listar Edificio Producción

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios de producción que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los edificios de producción registrados en el sistema.

RF40. Eliminar Edificio Producción

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios de producción que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del edificio de producción seleccionado.

Salida: Muestra el listado de los edificios de producción registrados en el sistema actualizado.

RF41. Buscar Edificio Producción

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios de producción que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del edificio de producción seleccionado.

Salida: Muestra los datos del edificio de producción seleccionado.

RF42. Adicionar Edificio Residencia

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo edificio de residencia.

Entrada: Datos del edificio de residencia que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El edificio de residencia queda adicionado en el sistema.

RF43. Modificar Edificio Residencia

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del edificio de residencia en cuestión.

Entrada: Datos del edificio de residencia que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el edificio de residencia actualizado.

RF44. Listar Edificio Residencia

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios de residencia que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los edificios de residencia registrados en el sistema.

RF45. Eliminar Edificio Residencia

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios de residencia que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del edificio de producción seleccionado.

Salida: Muestra el listado de los edificios de residencia registrados en el sistema actualizado.

RF46. Buscar Edificio Residencia

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los edificios de residencia que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del edificio de residencia seleccionado.

Salida: Muestra los datos del edificio de residencia seleccionado.

RF47. Adicionar Servicio

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo servicio.

Entrada: Datos del servicio que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El servicio queda adicionado en el sistema.

RF48. Modificar Servicio

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del servicio en cuestión.

Entrada: Datos del servicio que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el servicio ya actualizado.

RF49. Listar Servicio

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los servicios que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los servicios registrados en el sistema.

RF50. Eliminar Servicio

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los servicios que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del servicio seleccionado.

Salida: Muestra el listado de los servicios registrados en el sistema actualizado.

RF51. Buscar Servicio

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los servicios que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del servicio seleccionado.

Salida: Muestra los datos del servicio seleccionado.

RF52. Adicionar Servicios Médicos

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo servicio médico.

Entrada: Datos del servicio médico que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El servicio médico queda adicionado en el sistema.

RF53. Modificar Servicios Médicos

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del servicio médico en cuestión.

Entrada: Datos del servicio médico que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el servicio médico ya actualizado.

RF54. Listar Servicios Médicos

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los servicios médicos que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los servicios médicos registrados en el sistema.

RF55. Eliminar Servicios Médicos

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los servicios médicos que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del servicio seleccionado.

Salida: Muestra el listado de los servicios médicos registrados en el sistema actualizado.

RF56. Buscar Servicios Médicos

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los servicios médicos que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del servicio médico seleccionado.

Salida: Muestra los datos del servicio médico seleccionado.

RF57. Adicionar Puntos de Venta

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo punto de venta.

Entrada: Datos del punto de venta que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El punto de venta queda adicionado en el sistema.

RF58. Modificar Puntos de Venta

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del punto de venta en cuestión.

Entrada: Datos del punto de venta que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el punto de venta ya actualizado.

RF59. Listar Puntos de Venta

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los puntos de venta que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los puntos de venta registrados en el sistema.

RF60. Eliminar Puntos de Venta

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los puntos de venta que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del punto de venta.

Salida: Muestra el listado de los puntos de venta registrados en el sistema actualizado.

RF61. Buscar Puntos de Venta

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar el lugar de interés que está registrado en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del lugar de interés seleccionado.

Salida: Muestra los datos del lugar de interés seleccionado.

RF62. Adicionar Lugares de Interés

Descripción: El sistema a través de una interfaz de usuario debe permitir introducir los datos solicitados para crear un nuevo lugar de interés.

Entrada: Datos del lugar de interés que se va a adicionar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El lugar de interés queda adicionado en el sistema.

RF63. Modificar Lugares de Interés

Descripción: El sistema a través de una interfaz de usuario debe permitir modificar los datos del lugar de interés en cuestión.

Entrada: Datos del lugar de interés que se va a modificar (nombre, coordenadas, teléfonos, administrador, detalles, localización).

Salida: El sistema muestra un listado con el lugar de interés ya actualizado.

RF64. Listar Lugares de Interés

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los lugares de interés que están registrados en el sistema.

Entrada: No procede.

Salida: Muestra el listado de los lugares de interés registrados en el sistema.

RF65. Eliminar Lugares de Interés

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los lugares de interés que están registrados en el sistema, para seleccionar uno y eliminarlo.

Entrada: Identificador del lugar de interés seleccionado.

Salida: Muestra el listado de los lugares de interés registrados en el sistema actualizado.

RF66. Buscar Lugares de Interés

Descripción: El sistema a través de una interfaz de usuario debe permitir visualizar los lugares de interés que están registrados en el sistema, una vez seleccionado uno, debe permitir mediante una interfaz visualizar los datos de este.

Entrada: Identificador del lugar de interés seleccionado.

Salida: Muestra los datos del lugar de interés seleccionado.

2.2.2. Requisitos no funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema (21).

➤ Requisitos de software

Para las PCs cliente:

RNF1. Sistema operativo: GNU/Linux, Windows y Mac OS.

RNF2. Un Navegador como Mozilla Firefox, Chrome, Safari u otro navegador que sea capaz de interpretar el código JavaScript v1.4.1 o superior y cuente con soporte para CSS3 y HTML 5.

➤ Para los servidores:

RNF3. Sistemas Operativos GNU/Linux o Windows Server 2000 o superior.

RNF4. Servidor Web Apache 2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.

RNF5. PostgreSQL como Sistema Gestor de Base de Datos.

RNF6. PostGIS v1.5 como extensión de PostgreSQL y soporte de datos espaciales.

➤ Requisitos de Hardware

Para las PCs clientes:

RNF7. Mínima capacidad para la Memoria de Acceso Aleatorio (RAM por sus siglas en inglés *Random Access Memory*) debe ser de 512 MB.

RNF8. Procesador a 2.0 GHz como mínimo.

Para los servidores:

RNF9. El Servidor de BD tenga como mínimo 2GB de RAM y 10GB de disco duro libre.

RNF10. Procesador 3.0 GHz como mínimo.

➤ Requisitos de Apariencia o Interfaz de Usuario

RNF11. El sistema debe proveer de forma ordenada y detallada las funcionalidades del sistema.

RNF12. Los nombres de las funcionalidades principales del sistema estarán acompañados por íconos para un mayor reconocimiento visual por parte del usuario.

RNF13. El sistema debe tener un diseño sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para ser utilizado.

➤ **Requisitos de Usabilidad**

RNF14. El sistema está concebido para ser usado por usuarios autorizados de la universidad con conocimientos del negocio.

RNF15. El sistema será utilizado por cualquier miembro del equipo de desarrollo con los conocimientos suficientes para trabajar en él en futuras actualizaciones.

➤ **Requisitos de Eficiencia**

RNF16. El tiempo de respuesta y la velocidad de procesamiento estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento de actualización y recuperación.

➤ **Requisitos de Portabilidad**

RNF17. El sistema debe ser implementado de tal forma que pueda ser ejecutado en el sistema operativo Windows o Linux.

➤ **Requisitos de Escalabilidad**

RNF18. El sistema debe ser escalable de modo que futuras funcionalidades puedan ser implementadas e incorporadas. Por ejemplo, al sistema hoy solo pueden acceder los usuarios registrados en la base de datos, pero puede agregársele la funcionalidad de que se autenticquen por LDAP sin que este sufra afectaciones.

➤ **Requisitos de Seguridad**

RNF19. La información manejada por el sistema estará protegida de cualquier acceso no autorizado, debido a que los usuarios para acceder al sistema deben autenticarse para garantizar que cada usuario tenga acceso solo a lo que se le permite en dependencia del rol que se le esté asignado.

2.3. Modelo de casos de uso

Describe los procesos de un negocio vinculados al campo de acción, y cómo se benefician e interactúan los socios y clientes en estos procesos.

2.3.1 Actores del Sistema

Para el sistema en cuestión, los actores son personas que interactúan con la aplicación. Los actores poseen un rol que es determinado por los casos de uso al que tiene acceso, en este caso se identificaron dos actores principales: usuario y administrador. Este último es una especialización del primero, ya que comparte sus casos de uso y tiene otros que son específicos del mismo rol.

Tabla 2 Descripción de los actores.

Nombre del actor	Descripción
Usuario	Actor del sistema que gestiona toda la información existente, excepto la de gestionar usuario.
Administrador	Actor del sistema que gestiona toda la información existente.

2.3.2. Diagrama de caso de uso del sistema

El diagrama de casos de uso representa la forma de como un cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan. Documentan el comportamiento de un sistema desde el punto de vista del usuario.

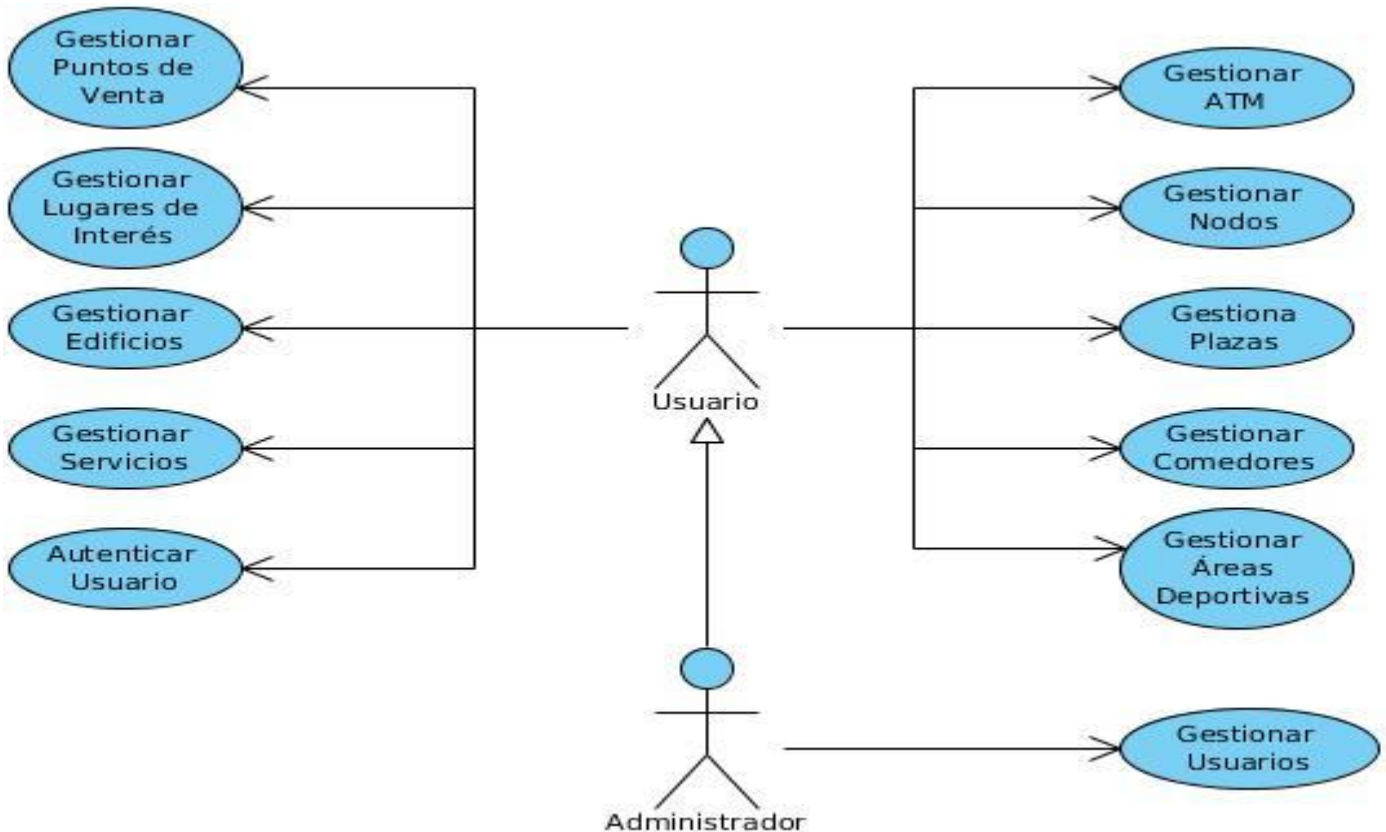


Figura. 3 Diagrama de caso de uso del sistema

2.3.3. Descripción de los casos de uso

Autenticar usuario: este caso de uso permite realizar el proceso de autenticación para acceder al sistema, en la que el usuario para poder acceder deberá introducir sus credenciales y si este se encuentra registrado podrá acceder, una vez dentro podrá trabajar con el sistema en dependencia del rol que tenga, serán las funcionalidades a las que tendrá acceso.

Gestionar servicios: este caso de uso permitirá listar, adicionar, editar, eliminar y buscar un servicio determinado ya sea de tipo servicio y servicio médico.

Gestionar edificios: los edificios son de tres tipos: docentes, producción y residencia. En este caso de uso el usuario autenticado tendrá la posibilidad de poder listar, adicionar, editar, eliminar y buscar un edificio respectivamente.

Gestionar lugares de interés: permite listar, adicionar, editar, eliminar y buscar un lugar de interés.

Gestionar puntos de venta: permite listar, adicionar, editar, eliminar y buscar un punto de venta.

Gestionar atm: permite listar, adicionar, editar, eliminar y buscar un atm.

Gestionar nodos: permite listar, adicionar, editar, eliminar y buscar un nodo.

Gestionar plazas: permite listar, adicionar, editar, eliminar y buscar una plaza.

Gestionar comedores: permite listar, adicionar, editar, eliminar y buscar un complejo comedor.

Gestionar áreas deportivas: permite listar, adicionar, editar, eliminar y buscar un área deportiva determinada.

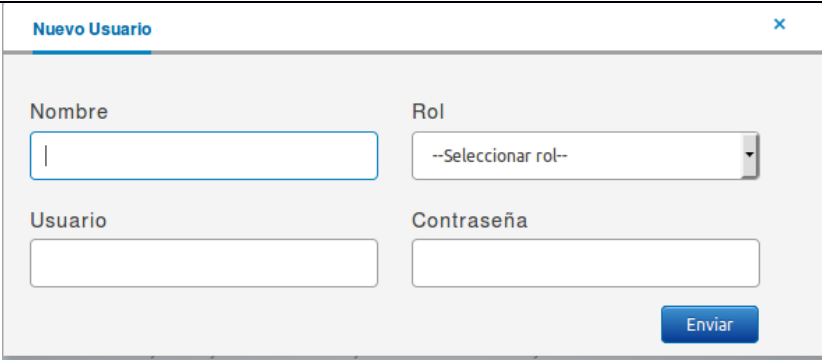
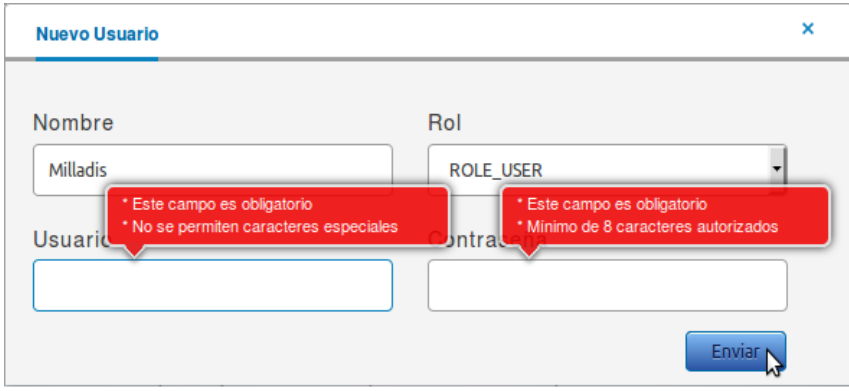
Gestionar usuarios: permite listar, adicionar, editar, eliminar y buscar un usuario determinado.

2.3.4. Descripción del Caso de Uso arquitectónicamente significativo del sistema

CU: Gestionar usuario.

Tabla 3 Descripción del caso de uso Gestionar Usuario

Caso de uso:	Gestionar usuario
Actores:	Administrador
Resumen:	En principio el administrador selecciona una de las funcionalidades con que cuenta el caso de uso (adicionar, editar, modificar, eliminar y buscar). El caso de uso termina cuando se ejecuta de forma satisfactoria alguna de las funcionales mencionadas anteriormente.
Precondiciones:	El sistema debe estar instalado y funcionando correctamente. Además el usuario debe estar autenticado con el rol de ROLE_ADMIN.
Referencias:	RF2, RF3, RF4, RF5, RF6
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción de Autor	Respuesta del Sistema
1. El actor selecciona la opción correspondiente al listado de usuarios.	2. El sistema muestra la lista de los usuarios registrados en el sistema a los que solo tiene acceso el administrador, el cual puede: <ul style="list-style-type: none"> - Adicionar usuario, ver sección Adicionar usuario. - Editar usuario, ver sección Editar usuario. - Eliminar usuario, ser sección Eliminar usuario. - Buscar usuario, ver sección Buscar usuario.

Sección "Adicionar usuario"	
Acción del actor	Respuesta del sistema
1. El administrador selecciona la opción "Nuevo"	2. El sistema le muestra al usuario la interfaz para insertar los datos necesitados para adicionar un usuario nuevo (Nombre, Usuario, Contraseña y Rol).
3. El administrador introduce los datos requeridos.	4. El sistema valida los datos insertados (El nombre debe ser de la forma Nombre - Primer Apellido -Segundo Apellido, se debe comenzar con letra mayúscula, el usuario debe ser de no más de 10 caracteres y no admite caracteres especiales, la contraseña debe tener no menos de 8 caracteres, el rol esta predefinido que son ROLE_USER y ROLE_ADMIN.)
5. El administrador selecciona la opción "Enviar".	6. El sistema adiciona el usuario.
Prototipo de interfaz	
	
Flujo alternativo del paso 4 "Validación"	
Acción del Actor	Respuesta del Sistema
	4a. El sistema muestra los campos que tienen errores con mensajes de color rojo en el que se puede ver el tipo de error que se cometió.
Prototipo de interfaz	
	
Flujo Alternativo al paso 5 "Cancelar"	

Acción del Actor	Respuesta del Sistema
5a. El administrador decide cancelar la operación en curso.	5b. El sistema cierra la interfaz

Prototipo de Interfaz

Sección "Editar usuario"

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción "Editar" del usuario que quiera modificar.	2. El sistema muestra una interfaz en la que se puede ver los datos del usuario a modificar.
3. El administrador procede a modificar los datos mostrados en la interfaz.	4. El sistema espera a que el administrador presione el botón "Editar".
5. El administrador presiona el botón "Editar".	6. El sistema valida los datos entrados por el administrador, envía los datos y muestra el listado de usuarios actualizados terminando así el caso de uso.

Prototipo de Interfaz

Flujo Alternativo al paso 6 "Validación"

Acción del Actor	Respuesta del Sistema

	6 a. El sistema muestra un mensaje de error en los campos que no cumplan con la validación requerida.
--	---

Prototipo de Interfaz

Flujo Alternativo al paso 5 "Cancelar"

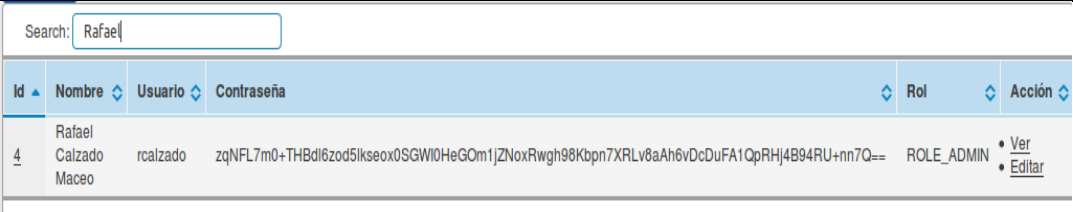
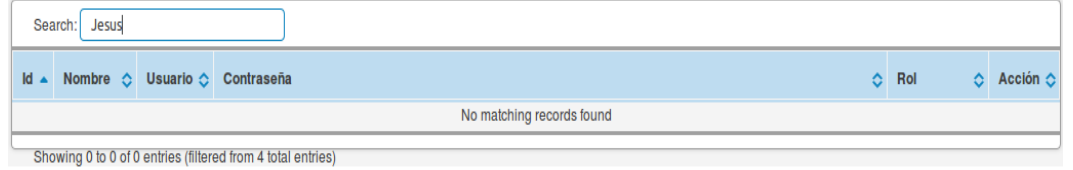
Acción del Actor	Respuesta del Sistema
5 a. El administrador desiste de efectuar la operación de editar y presiona el botón cancelar.	5 b. En caso de que el administrador no haya modificado ningún campo entonces se cierra la interfaz de editar usuario y muestra el listado de usuarios registrados.

Prototipo de Interfaz

Sección "Eliminar usuario"

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción "Editar" del usuario a eliminar.	2. El sistema muestra la interfaz de confirmación, preguntando al usuario si desea continuar.
3. El administrador presiona el botón "Aceptar".	4. El sistema elimina el usuario.

Prototipo de interfaz

<div style="border: 1px solid #ccc; padding: 10px;"> <p style="text-align: right; margin: 0;">Datos del Usuario ×</p> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <div style="width: 45%;"> <p>Nombre</p> <input type="text" value="Yolaine Pérez Hernández"/> </div> <div style="width: 45%;"> <p>Rol</p> <input type="text" value="ROLE_USER"/> </div> </div> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <div style="width: 45%;"> <p>Usuario</p> <input type="text" value="yperez"/> </div> <div style="width: 45%;"> <p>Contraseña</p> <input type="text" value="M4hHGu9Z7qa5IApEeKKRqdMtg//EJh+Qi"/> </div> </div> <div style="text-align: right; margin-top: 10px;"> <input type="button" value="Eliminar"/> <input type="button" value="Editar"/> </div> </div>	
Sección “Buscar usuario”	
Acción del Actor	Respuesta del Sistema
1. El usuario se posiciona en el campo buscar que se encuentra en la parte superior izquierda de la interfaz donde se muestra el listado de los usuarios registrados y el criterio de búsqueda.	2. El sistema muestra el resultado según el criterio de búsqueda introducido por el usuario.
Prototipo de Interfaz	
 <p>The screenshot shows a search bar with the text 'Rafael'. Below it is a table with columns: Id, Nombre, Usuario, Contraseña, Rol, and Acción. The table contains one row for 'Rafael Calzado Maceo' with role 'ROLE_ADMIN' and actions 'Ver' and 'Editar'.</p>	
Flujo Alternativo al paso 2 “Error al buscar”	
Acción del Actor	Respuesta del Sistema
	2 a. En caso de que no existe ningún usuario que coincida con el criterio de búsqueda, el sistema mostrará un mensaje de que no se encontraron resultados.
Prototipo de Interfaz	
 <p>The screenshot shows a search bar with the text 'Jesus'. Below it is a table with columns: Id, Nombre, Usuario, Contraseña, Rol, and Acción. The table is empty, and a message 'No matching records found' is displayed. At the bottom, it says 'Showing 0 to 0 of 0 entries (filtered from 4 total entries)'.</p>	
Pos-condiciones	El sistema queda con un nuevo usuario creado, modificado o eliminado de los existentes.

2.4. Modelo de diseño

El Modelo de Diseño es una abstracción del modelo de implementación de un sistema y se utiliza como una entrada fundamental en este proceso. Su objetivo es especificar una solución gráfica que pueda ser

convertida posteriormente en código fuente. Para cumplir su objetivo se puede emplear el lenguaje de modelado UML para la confección de diagramas de paquetes y de clases (22).

2.4.1. Diagrama de paquetes

La organización de las clases en paquetes ofrece la ventaja de los elementos detallados en abstracciones más amplias, lo cual proporciona soporte a una vista superior y permite contemplar el modelo en agrupamientos más simples. Este tipo de diagrama se centra en organizar los paquetes presentes en la aplicación y sus respectivos elementos. Para la organización del código fuente de la aplicación Symfony 2.3 propone una estructura predefinida de directorios. En la siguiente figura se muestra el diagrama de paquetes para el Sistema de Administración para el SIGUCI.

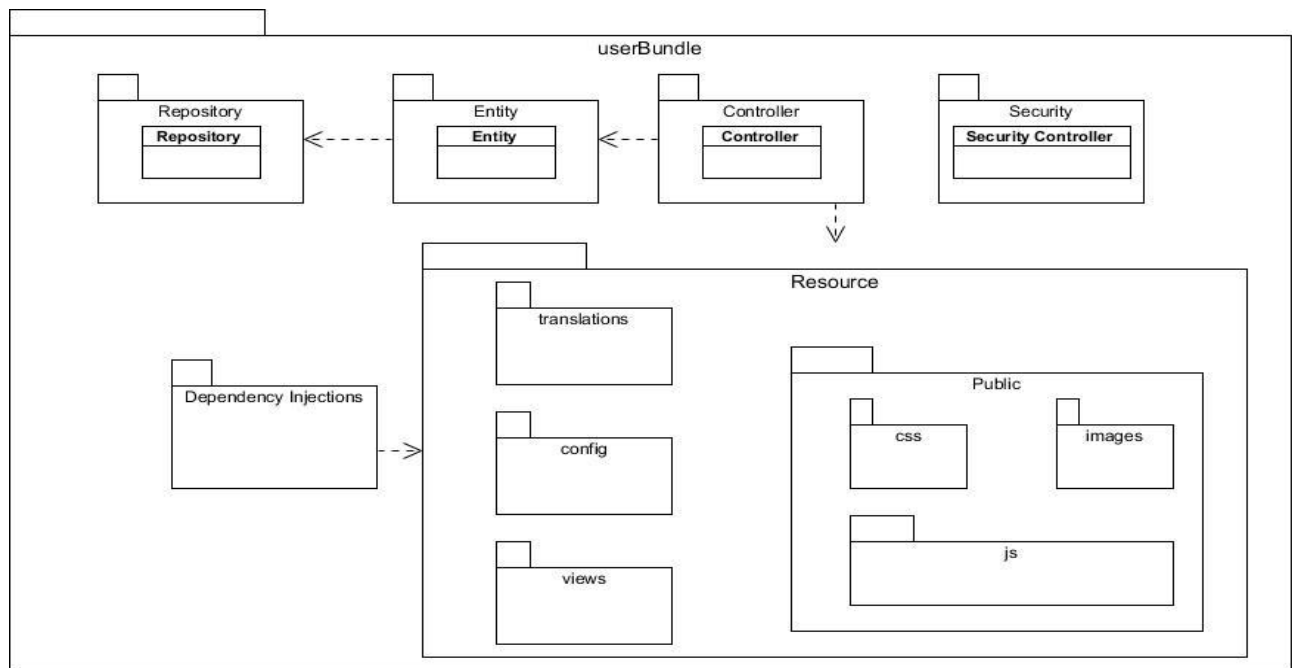


Figura. 4 Diagrama de paquetes SIGADMIN

El directorio Controller contendrá las clases controladoras del sistema. Estas son de manera general las responsables de la gestión del sistema para la administración del SIGUCI. Para el cumplimiento de dicha responsabilidad cuenta en el directorio Entity que alberga todas las entidades, el directorio Repository y los archivos públicos que están en el directorio Resource. Las entidades son las clases encargadas de la abstracción de la lógica relacionada con los datos, proporcionando el uso de los mismos en dependencia del gestor de base de datos utilizado. El directorio *Repository* contiene las clases con las sentencias DQL

(*Doctrine Query Lenguaje*) propio de *Doctrine*. El directorio *Dependency Injections* en *Symfony 2.3* por defecto contiene los archivos necesarios para el correcto funcionamiento del Inyector de Dependencias. Por último esta e directorio *Resource* que contiene las clases de la vista así como la carpeta *public* que contiene todos los archivos públicos del sistema.

2.4.2. Diagramas de Clases del Diseño

El diagrama de Clases del Diseño describe gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación (23).

2.4.3. Diagrama de Clases del Diseño del caso de uso Gestionar usuario.

En el siguiente diagrama de clases del diseño se muestran las principales clases con los respectivos métodos y atributos que contienen, para el caso de uso gestionar usuario.

En el diagrama de clases para el caso de uso Gestionar Usuario la clase *CP_Usuario* es la interfaz con la que interactúa el administrador a la hora de adicionar un nuevo usuario, esta a su vez usa la librería *Jquery* para la personalización de la interfaz. Además contiene un formulario por donde se envían los datos por submit a la *SP_App*, la misma está relacionada con la *SP_AppKernel* donde se registran todos los bundles de la aplicación, en este caso contiene el *userBundle* que contiene la clase *UsuarioController*, esta es la clase que contiene todas la operaciones que se realizan con los usuarios utilizando la librería *Doctrine* para conectarse a la base de datos.

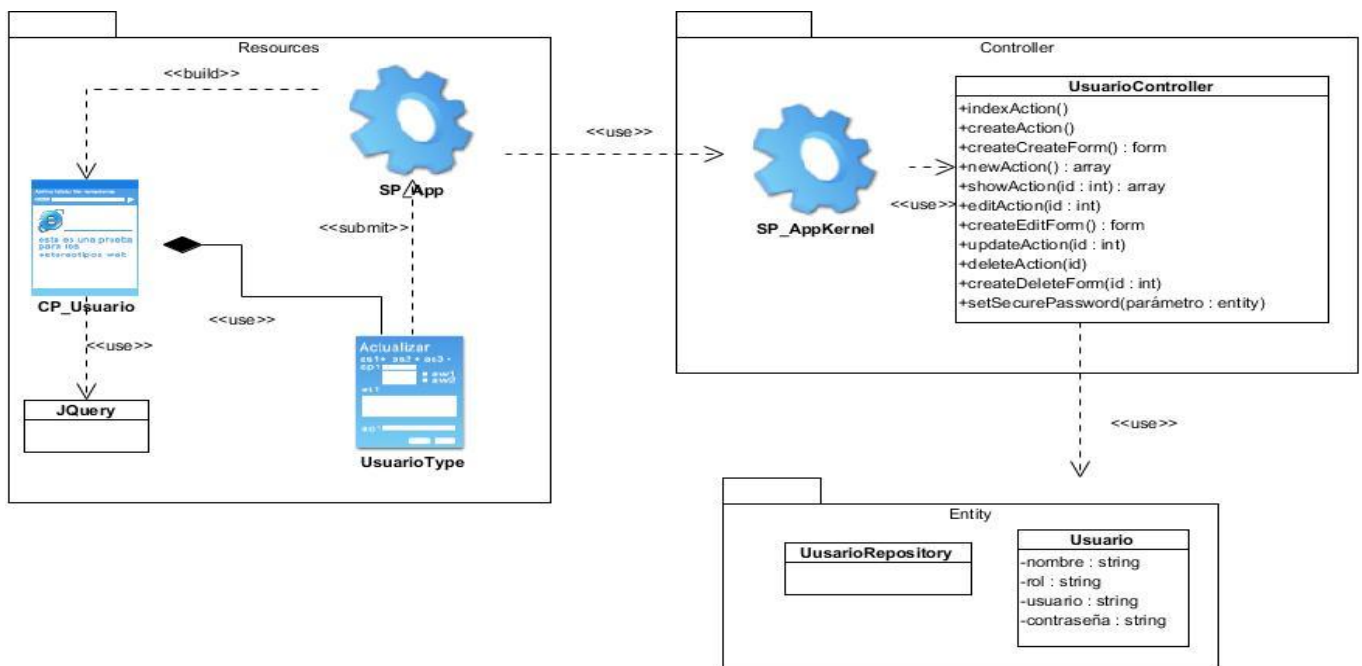


Figura. 5 Diagrama de clases del diseño del caso de uso Gestionar usuario.

2.5. Patrones utilizados para el desarrollo de la solución

Un patrón de diseño consiste en un diagrama de objetos que forma una solución a un problema conocido y frecuente. El diagrama de objetos está constituido por un conjunto de objetos descritos por clases y las relaciones que enlazan los objetos (24).

2.5.1. Patrones de casos de uso

- **CRUD Completo**

Para el diseño del sistema se utilizó el patrón de caso de uso CRUD Completo permitiendo modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja. Este patrón deberá ser usado cuando todas las operaciones contribuyen al mismo valor de negocio y todas son cortas y simples. Por ejemplo en el caso de uso gestionar usuario para efectuar las operaciones de adicionar, editar, eliminar y ver.

- **Múltiples Actores Rol Común:**

Este patrón indica que dos actores juegan el mismo rol sobre un caso de uso. Evidenciado en la relación de herencia entre Usuario y Administrador.

2.5.2. Patrones Arquitectónicos

Son los que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Se seleccionó el patrón modelo-vista-controlador (MVC) basado en el marco de trabajo del framework Symfony2 el cual fue utilizado en la solución del sistema. A continuación se explica en que consiste dicho patrón.

- **MVC**

El MVC es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). En la figura 5 se evidencia en el paquete Entity.

Vista: La vista es la encargada de originar las páginas que son mostradas como resultado de las acciones, donde se encuentra el layout, que es común para todas las páginas de la aplicación. En la figura 5 se evidencia en el paquete Resource.

Controlador: Responde a eventos e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información, por tanto se podría decir que el controlador hace de intermediario entre la vista y el modelo. En la figura 5 se evidencia en el paquete Controller.

2.5.3. Patrones de diseño

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. Los patrones de arquitectura expresan un esquema organizativo estructural fundamental para sistemas de software.

Patrones GRASP

- Creador

El propósito del patrón creador es encontrar un creador que tenga que conectarse con el objeto producido en cualquier evento. En la figura 6 se evidencia el empleo de este patrón. La clase controladora UsuarioController es la responsable de crear instancias de las entidades de tipo Usuario.

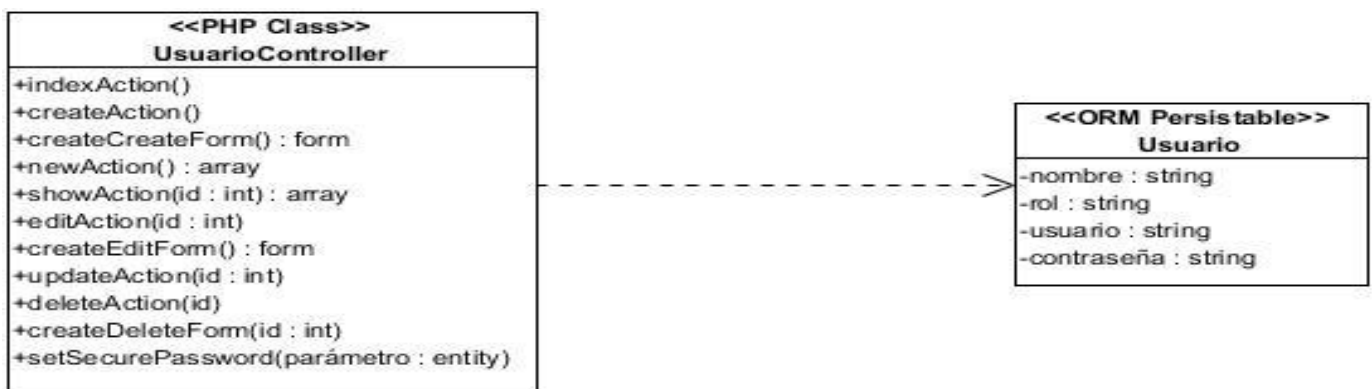


Figura. 6 Evidencia del patrón creador

- **Controlador**

Este patrón sugiere que la lógica del negocio esté separada de la capa de presentación. Symfony garantiza el empleo de este patrón con la definición de clases controladoras por cada entidad del sistema. Cada una de estas clases posee responsabilidades específicas para controlar el flujo de eventos del sistema. La figura 6 contiene la clase UsuarioController que es la encargada de manejar el flujo de eventos asociados a la gestión de usuarios en el sistema.

- **Alta cohesión**

Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase UsuarioController y Usuario. La clase controladora UsuarioController contiene los métodos necesarios para la gestión de usuarios de manera general, pero por ejemplo para llevar a cabo la operación de inserción hace uso de la clase Usuario que contiene los métodos para la gestión de cada uno de los atributos de la clase en la base de datos.

- **Experto**

Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Propel para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan. Este patrón queda evidenciado en la entidad Usuario del paquete del modelo mostrado en la figura 5.

Patrones GoF

- **Patrón Instancia Única**

Clase sfRouting – método getInstance. Esta clase la utiliza el controlador frontal (sfWebFrontController) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. La instancia única sfRouting precisa otros métodos muy útiles para la gestión manual de las rutas: ClearRoutes (), hasRoutes (), getRoutesByName ().

- **Patrón Decorador**

Este método pertenece a la clase abstracta `sfView`, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado `base.html.twig` es el que contiene el Layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este procedimiento es una implementación del patrón Decorator.

2.6. Modelo de Datos

Un modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento (25).

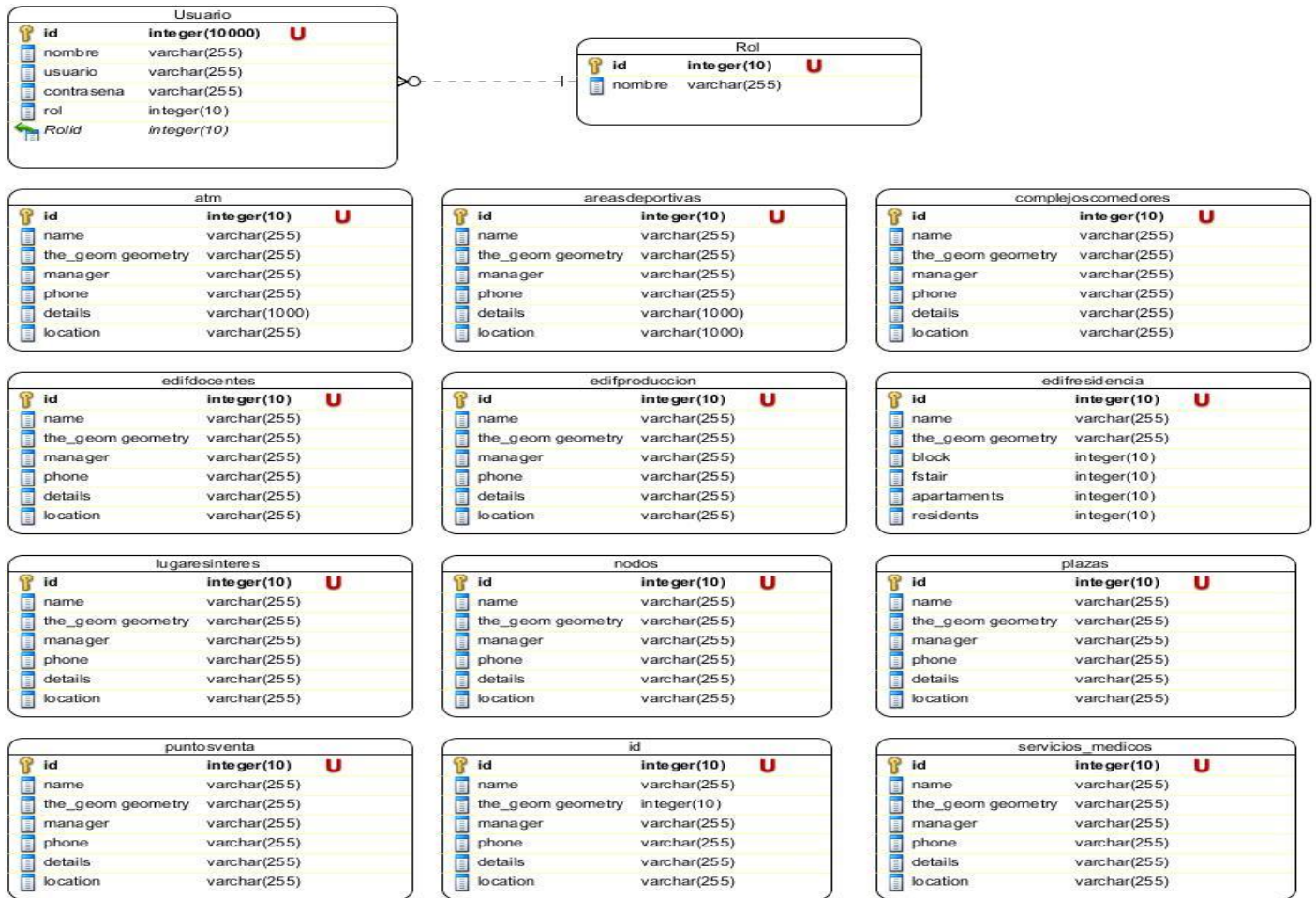


Figura. 7 Modelo de Datos

2.6.1. Descripción de los principales objetos del modelo de datos

Tabla 4 Descripción del objeto usuario.

Nombre: usuario		
Descripción: Guarda los datos de los usuarios registrados.		
Atributo	Tipo	Descripción
id	integer(10)	Identificador auto-incremental.
nombre	varchar(255)	Nombre del usuario.
usuario	varchar(255)	Usuario para la autenticación

contraseña	varchar(255)	Contraseña para acceder al sistema.
rol	integer(10)	Rol que tiene el usuario.

Tabla 5 Descripción del objeto nodos.

Nombre: Nodos		
Descripción: Guarda los datos de los nodos guardados en la base de datos.		
Atributo	Tipo	Descripción
id	integer(10)	Identificador auto-incremental.
nombre	varchar(255)	Nombre del nodo.
Coordenadas	varchar(255)	Ubicación en la universidad
Administrador	varchar(255)	Persona que administra el nodo.
Teléfono	integer(10)	Número de teléfono en la instalación.
Localización	varchar(255)	Breve descripción de la ubicación del nodo.
Detalles	varchar(255)	Otros datos referentes al nodo.

Tabla 6. Descripción del objeto rol.

Nombre: rol		
Descripción: Roles de los usuarios.		
Atributo	Tipo	Descripción
id	integer(10)	Identificador auto-incremental.
nombre	varchar(255)	Nombre del rol.

2.7. Modelo de despliegue

El diagrama de despliegue permite mostrar la arquitectura en tiempo de ejecución del sistema respecto a *hardware* y *software*. El diagrama de despliegue se utiliza en el diseño y la implementación. Se pueden distinguir componentes y nodos, así como la relación entre todos estos (26).



Figura. 8 Modelo de despliegue

En el modelo anterior se encuentran representados los nodos PC Cliente que tendrá instalado un navegador web con soporte para JavaScript y a su vez estará conectado al servidor web por el protocolo de transferencia de hipertexto [*Hypertext Transfer Protocol Secure*] (HTTPS), el nodo Servidor WEB que tendrá instalado un servidor web y está conectado al servidor de base de datos por el protocolo de comunicación TCP/IP.

Conclusiones parciales

En el presente capítulo fueron identificados 66 requisitos funcionales y 19 requisitos no funcionales permitiendo definir las funcionalidades y características que debe cumplir el sistema. La elaboración de los diagramas de clases del diseño y los diagramas de secuencia propició una mejor comprensión de la distribución de las clases, así como de las relaciones y responsabilidades de cada una de ellas.

La aplicación adecuada de los patrones de diseño permitió asignar las responsabilidades correspondientes a cada clase. Además en el diagrama de casos de uso del sistema se identificaron los actores del sistema y los casos de uso relacionados a ellos. Para representar el entorno físico de cómo se vería la solución se realizó el diagrama de despliegue.

CAPÍTULO III: “IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA DE ADMINISTRACIÓN PARA EL SIGUCI”

En el presente capítulo se muestra el diagrama de componentes referente a la seguridad del sistema, los estándares y estilos de codificación utilizados, se describen las pruebas a utilizar con el objetivo de encontrar los problemas aún existentes para de esta forma poder corregirlos, contribuyendo así al mejoramiento de manera general del sistema.

3.1. Modelo de implementación

La implementación brinda la posibilidad de probar y desarrollar componentes como unidades que finalmente serán integrados como un sistema ejecutable. En este flujo de trabajo se organizan y realizan las pruebas unitarias y se integran los componentes implementados, basándose en las especificaciones de diseño.

3.1.1. Diagrama de componentes

El diagrama de componentes describe la descomposición física del sistema de software en componentes, a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos (26).

En la siguiente figura se muestra el diagrama de componentes para el caso de uso “Gestionar Usuarios” el cual está conformado por 3 paquetes de implementación básicos, estos son:

- Paquete de Clases Modelo: este paquete reúne las clases que interactúan con la base de datos.
- Paquete de Clases Vista: este es el paquete encargado de agrupar los componentes que permiten la interacción directa del usuario final con el sistema.
- Paquete de Clases Controlador: contendrá las clases que manipulan las acciones del usuario y se apoya en el subsistema del modelo para dar respuesta a las peticiones de la vista.

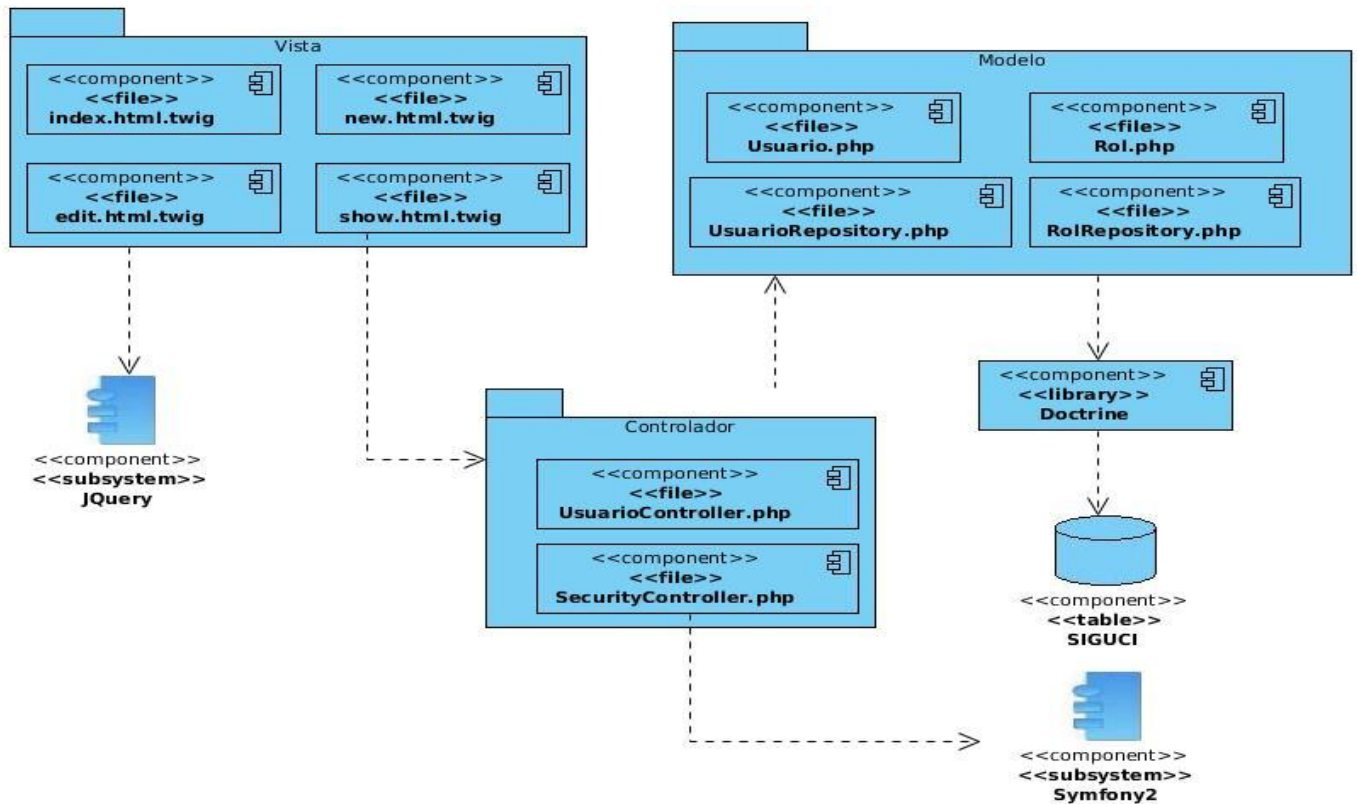


Figura. 9 Diagrama de componentes

3.1.2. Código fuente

El código fuente, o *source code*, también llamado código base, es un texto que se ha escrito en un lenguaje de programación concreto, y que sólo puede ser leído por un experto o programador. Estos caracteres deben ser traducidos a un lenguaje que se denomina código máquina, el cual podrá ejecutar cualquier ordenador. También puede ser traducido a un lenguaje llamado códigos de bytes, el cual podrá ser traducido por un intérprete. Este tipo de transferencias y traducciones se llaman compilación. (Raul Bonenfant Muñiz, 2014). En la figura 11 hay un ejemplo de código fuente donde se hace uso de los estándares de codificación.

3.1.3. Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador

hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, establezca un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente. (Fabien Potencier, 2009).

3.1.4. Estilos de codificación utilizados

- Los nombres de las variables serán con minúscula, en caso de ser compuestas se escribirán juntas.
- Los métodos se separarán entre sí por una línea en blanco.
- Las variables locales se inicializarán en el momento de su creación y estas deben situarse al principio de cada bloque principal en el que se utilicen y no en el momento de su uso.
- Los métodos comenzarán con letra inicial minúscula, en caso de ser compuesto usarán el estilo de escritura lowerCamelCase.

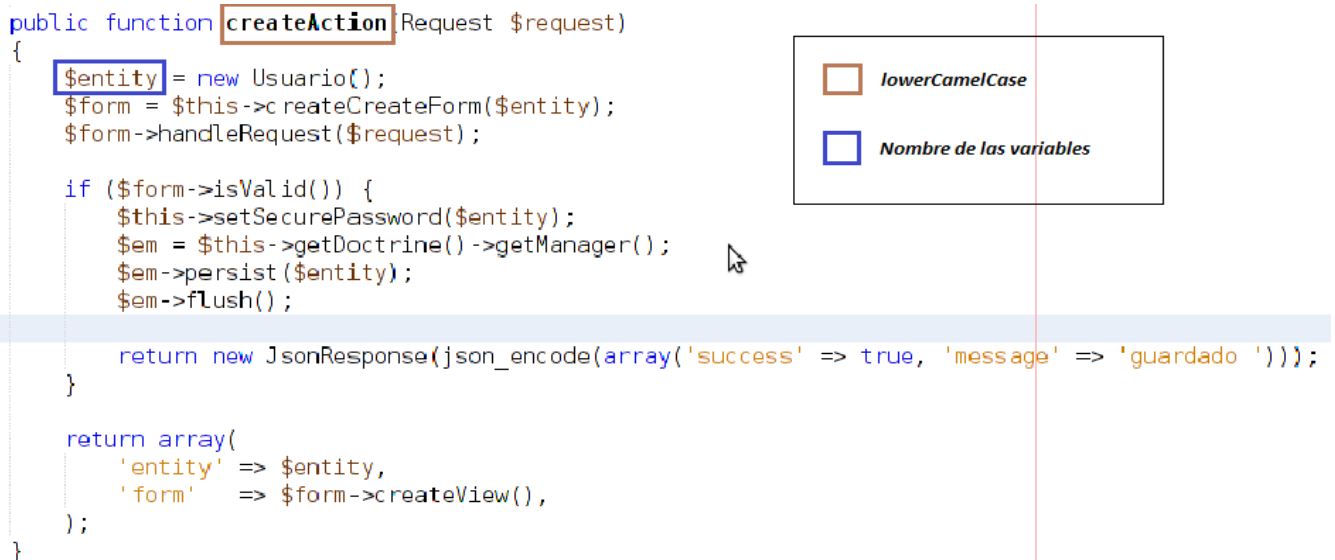
Ejemplo de código fuente

```
public function createAction(Request $request)
{
    $entity = new Usuario();
    $form = $this->createCreateForm($entity);
    $form->handleRequest($request);

    if ($form->isValid()) {
        $this->setSecurePassword($entity);
        $em = $this->getDoctrine()->getManager();
        $em->persist($entity);
        $em->flush();

        return new JsonResponse(json_encode(array('success' => true, 'message' => 'guardado ')));
    }

    return array(
        'entity' => $entity,
        'form' => $form->createView(),
    );
}
```





	lowerCamelCase
	Nombre de las variables

Figura. 10 Ejemplo de empleo del estándar lowerCamelCase

3.2. Pruebas del software

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones del diseño y de la codificación. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las prueba para la detección de errores (27).

3.2.1. Niveles de prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de prueba:

- Prueba de Desarrollador
- Prueba Independiente
- Prueba de Unidad
- Prueba de Integración
- Prueba de Sistema
- Prueba de Aceptación

En el desarrollo de la fase de pruebas en función de que se cumplan los requisitos funcionales descritos con anterioridad se aplicaron los siguientes tipos de pruebas:

Pruebas funcionales: prueban una funcionalidad completa donde pueden estar implicadas una o varias clases y la propia interfaz de usuario.

Pruebas de Seguridad: tiene como principal objetivo verificar antes de la liberación del sistema, la aplicación de los mecanismos de protección incorporados y se realizan para detectar la consecuente existencia de vulnerabilidades y/o defectos de seguridad para eliminarlos.

3.2.2. Método de Prueba

Pruebas de Caja Negra: se llevan a cabo sobre la interfaz del software y pretenden demostrar que el software funciona adecuadamente; es decir, que las entradas se aceptan de forma adecuada y que se

produce una salida correcta. Estas pruebas no tienen en cuenta la estructura lógica interna del software (27).

3.2.3. Diseño de Casos de Prueba

Conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. Siempre es ejecutada como una unidad, desde el comienzo hasta el final. Cada caso de uso debe estar asociado un caso de prueba dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión.

En la siguiente tabla se detallan las variables que se encuentran asociadas al caso de uso Gestionar Usuario.

Tabla 7 Descripción de las variables del caso de uso "Gestionar Usuario".

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Campo obligatorio. Formato: Alfanumérico. Mayúscula al inicio y después de cada espacio. Solo acepta letras y espacios.
2	Usuario	Campo de texto	No	Campo obligatorio. No acepta caracteres especiales.
3	Rol	Campo de selección	No	Campo obligatorio.
4	Contraseña	Campo de texto	No	Campo obligatorio. Debe tener no menos de 8 caracteres para que sea válido.

La descripción de variables anteriormente realizada permitió que se realizara una matriz de datos donde se evaluaron y probaron la validez de los datos introducidos en el sistema, específicamente el caso de uso "Gestionar Usuario" utilizando datos válidos e inválidos, identificando el empleo de la técnica de partición de equivalencia. Esta técnica se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de datos válidos e inválidos para condiciones de entrada.

Matriz de Datos

Tabla 8 Escenario "Adicionar Usuario"

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	V1	V2	V3	V4	Flujo central
SC1: Adicionar Usuario	EC 1.1: Adicionar usuario satisfactoriamente	El sistema adiciona el usuario mostrando un mensaje de confirmación.	V (Benita Maceo Ruiz)	V (bruiz)	V (ROLE_USER)	V (benita12345)	<ol style="list-style-type: none"> Clic en la opción "Usuarios" del menú. En la pestaña Usuarios clic en el botón "Nuevo". Rellenar los datos correspondientes. Clic sobre el botón "Enviar".
	EC 1.2: Se dejan sin rellenar campos que son obligatorios.	El sistema no envía los datos y muestra mensajes de color rojo por cada campo en blanco diciendo que son obligatorios.	I ()	V (bruiz)	I ()	V (benita12345)	<ol style="list-style-type: none"> Clic en la opción "Usuarios" del menú. En la pestaña Usuarios clic en el botón "Nuevo". Rellenar los datos correspondientes. Clic sobre el botón "Enviar".
			V (Benita Maceo Ruiz)	I ()	V (ROLE_USER)	I ()	
EC 1.3: Clic en la opción "Cerrar" de la ventana.	El sistema cierra la ventana	NA	NA	NA	NA	<ol style="list-style-type: none"> Clic en la opción "Usuarios" del 	

		cancelando así la operación en curso.					<ul style="list-style-type: none"> menú. 2. En la pestaña Usuarios clic en el botón "Nuevo". 3. Clic en la opción cerrar de la ventana.
	EC 1.4: Los datos introducidos no cumplen con el formato establecido.	El sistema no envía los datos y muestra mensajes de color rojo diciendo cuales fueron los errores cometidos.	V (Benita Maceo Ruiz)	V (bruiz)	V (ROLE_USER)	I (qwe)	<ul style="list-style-type: none"> 1. Clic en la opción "Usuarios" del menú. 2. En la pestaña Usuarios clic en el botón "Nuevo". 3. Rellenar los datos correspondientes. <p>Clic sobre el botón "Enviar".</p>

3.3. Resultados de las pruebas

3.3.1. Caja Negra

Después de realizar las pruebas de caja negra mediante los casos de prueba asociados a cada caso de uso, fueron realizadas un total de cuatro iteraciones detectando en la primera doce no conformidades, en la segunda iteración siete no conformidades y en la tercera, cinco no conformidades y una cuarta iteración donde no se detectaron no conformidades, comprobando que el sistema cumpliera con todos los requisitos funcionales y que los campos solo aceptaran los caracteres válidos para los mismos.

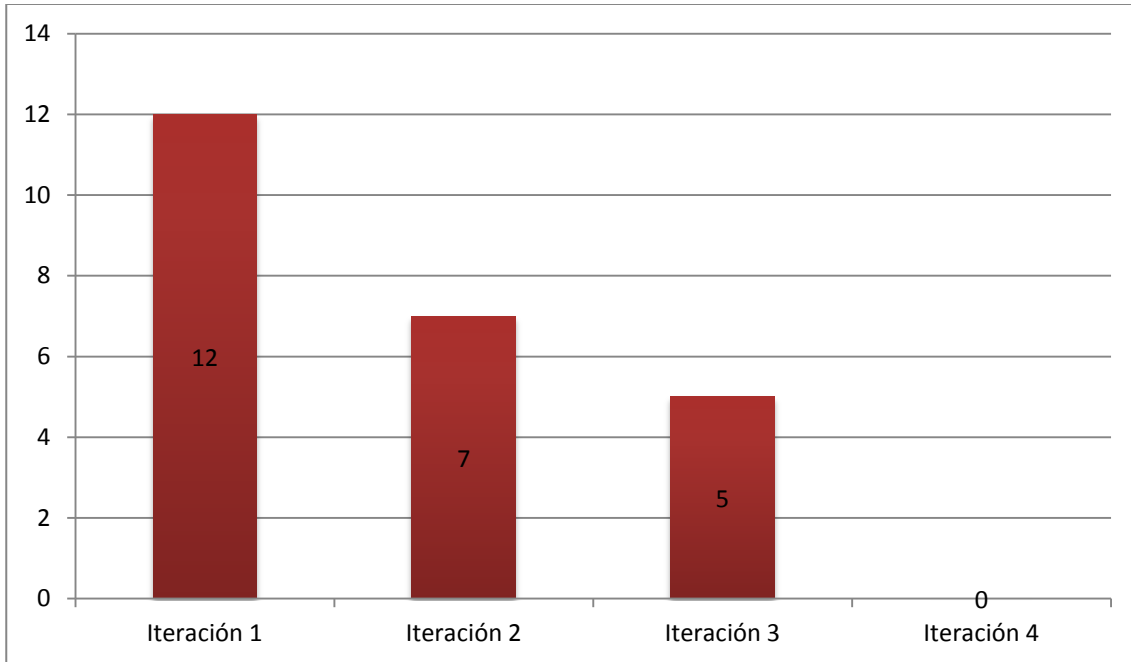


Figura. 11 Resultados de las pruebas

3.3.2. Pruebas de Seguridad y Control de Acceso

Con el objetivo de proporcionar una mayor seguridad en los datos del sistema se realizaron las pruebas de seguridad y control de acceso las cuales se centran en dos áreas claves de seguridad. Para el nivel de seguridad de la aplicación se verificó que un autor solo pueda tener acceso a las funciones y datos que su usuario tiene permitido, así como para la seguridad del sistema se verificó que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla.

Tabla 9 Descripción de las pruebas de seguridad y control de acceso

Casos de pruebas	
Descripción de la prueba	Ejemplo
Las pruebas de seguridad de la aplicación garantizan que, con base en la seguridad deseada, los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que están autorizados a acceder.	Cada usuario puede estar autorizado a modificar los datos existentes en la base de datos excepto los datos referentes a los usuarios registrados en el sistema.
Las pruebas de seguridad del sistema garantizan que solamente aquellos usuarios autorizados a	Luego de disímiles intentos violación el sistema respondió de manera favorable no dejando acceder al usuario de ninguna de las formas

acceder al sistema son capaces de ejecutar las funciones del sistema a través de los mecanismos apropiados. Las pruebas de seguridad del sistema garantizan que solamente aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema a través de los mecanismos apropiados. El objetivo de esta prueba es evaluar el funcionamiento correcto de los controles de seguridad del sistema para asegurar la integridad y confidencialidad de los datos	probadas, demostrando que el sistema cuenta con un mecanismo de autenticación seguro garantizando que al sistema puedan acceder solo aquellos usuarios que están registrados en el sistema.
---	---

3.3.3. Prueba de Usabilidad

La prueba de usabilidad se basa principalmente en determinar cuán bien el usuario podrá usar y entender la aplicación además de identificar las áreas de diseño que hacen al sistema de difícil uso para el usuario.

Para llevar a cabo esta prueba se utilizaron 5 usuarios a los cuales se les pidió interactuar con el sistema y ejecutar las funcionalidades para lo cual este fue diseñado en dependencia del rol especificado, mientras el equipo de desarrollo fue tomando nota de los errores que tenía el sistema en función de sus funcionalidades. Los resultados obtenidos fueron satisfactorios ya que los usuarios entendieron de manera fácil el funcionamiento del sistema, además de que presentaba una interfaz amigable y con una buena mensajería la cual les servía de guía a los usuarios para una mejor interacción con el sistema. Por último los usuarios involucrados en la prueba manifestaron el estar satisfechos con el funcionamiento del sistema.

Conclusiones parciales

En el desarrollo del presente capítulo se realizó el modelo de implementación de la parte de seguridad del sistema con el propósito de mostrar sus componentes y relaciones. Se especificó el uso de los estándares de codificación para lograr un estilo claro y organizado del código durante la fase de implementación. Se realizaron pruebas de seguridad, funcionales y de usabilidad para validar el correcto funcionamiento del módulo, utilizando el método de caja negra basado en la técnica de partición de equivalencia. En sentido general los resultados de las pruebas fueron productivos y satisfactorios, debido a que éstos arrojaron 12 no conformidades, las que fueron corregidas posteriormente, siendo muy importante para la satisfacción del cliente.

Conclusiones generales

Una vez concluida la investigación se puede afirmar que se cumplieron todos los objetivos planteados llegando a las siguientes conclusiones:

- La fundamentación de la metodología, herramientas y tecnologías empleadas en la construcción del sistema de administración para el SIGUCI permitió identificar el ambiente de desarrollo de acuerdo a las necesidades del proyecto.
- La realización del análisis y diseño y el estudio de los principales conceptos relacionados con el sistema para la administración del SIGUCI, permitió obtener como resultados los artefactos y diagramas necesarios para guiar el desarrollo de la aplicación.
- La implementación de las funcionalidades del Sistema de Administración para el SIGUCI dio paso al cumplimiento de los requisitos funcionales definidos para la solución.
- Con la realización de las pruebas de caja negra al Sistema para la Administración del SIGUCI se comprobó el correcto funcionamiento del software.

RECOMENDACIONES

Debido a la importancia que lleva el proceso de prueba para la corrección de errores en una aplicación. Se recomienda:

- Se recomienda para versiones futuras del sistema de administración para el SIGUCI, la gestión dinámica de nuevas entidades.

REFERENCIAS BIBLIOGRÁFICAS

1. Software SIG. *Software SIG*. [En línea] [Citado el: 15 de 01 de 2015.] <http://www.e-sig.info/software-sig/>.
2. esri. *esri*. [En línea] [Citado el: 15 de 01 de 2015.] <http://www.esri.es/>.
3. ArcGIS for Desktop. *ArcGIS for Desktop*. [En línea] [Citado el: 17 de 01 de 2015.] www.esri.com/software/arcgis/arcgis-for-desktop.
4. *Revista Bimestre Cubana*. **Silva, M Recio**. 2015.
5. **M, Ing. Antonio**. [En línea] 9 de 2009. <http://www.empai-matanzas.co.cu/revista/NUMERO%203%20diciembre%202008.pdf>.
6. **Gimson, Lic. Loraine**. Trabajo final integrador. *Trabajo final integrador*. [En línea] 9 de 2012. http://sedici.unlp.edu.ar/bitstream/handle/10915/24942/Documento_completo___pdf?sequence=1.
7. **Ríos Salgado, Santiago, Hinojosa Raza, Ing. Cecilia y Delgado Rodrí, Ing. Ramiro**. APLICACIÓN DE LA MET. *APLICACIÓN DE LA MET*. [En línea] 2013. <http://repositorio.espe.edu.ec/bitstream/21000/6316/1/AC-SISTEMAS-ESPE-047042.pdf>.
8. Unified Modeling Language™ (UML®) Resource Page. *Unified Modeling Language™ (UML®) Resource Page*. [En línea] <http://www.uml.org/>.
9. **Podeswa, Howard**. *PROGRAMACION UML*. s.l. : Anaya Multimedia, 2010, 2010.
10. NetBeans. *NetBeans*. [En línea] https://netbeans.org/index_es.html.
11. pgAdmin PostgreSQL Tools. [En línea] www.pgadmin.org/.
12. **Álvarez, Miguel Angel**. *desarrolloweb.com*. *desarrolloweb.com*. [En línea] 2001. <http://www.desarrolloweb.com/articulos/que-es-html.html>.
13. **Revuelta Domínguez, Francisco Ignacio y Pérez Sánchez, Lourdes**. *Interactividad de los entornos en la formación on-line*. s.l. : Editorial UOC, 2011.
14. **Muñoz, Vicente Javier Eslava**. *El nuevo PHP. Conceptos avanzados*. 2013.
15. MAESTROS DEL WEB. *MAESTROS DEL WEB*. [En línea] <http://www.maestrosdelweb.com/phpintro/>.
16. Federación nacional de Empresas de Software Libre. *Federación nacional de Empresas de Software Libre*. [En línea] <http://www.asolif.es/?q=content/http-apache-server>.
17. **Ruiz, N.** [En línea] 2011. http://webcache.googleusercontent.com/search?q=cache:tKRnaMj0RhcJ:repositorio.uta.edu.ec/jspui/bitstream/123456789/416/2/Tesis_t632si.pdf+&cd=1&hl=es&ct=clnk&gl=cu.

18. **Elespuru, A Baksai.** Software para la manipulación de Bases de Datos Espaciales. *Software para la manipulación de Bases de Datos Espaciales.* [En línea] 2007. http://dugi-doc.udg.edu/bitstream/handle/10256/1225/Software_Com.pdf?sequence=1.
19. **Álvarez, Miguel Angel.** *Manual de jQuery.*
20. **González Palacio, Liliana y Urrego Giraldo, Germán.** *Modelo de contexto y de dominio para la ingeniería de requisitos de sistemas ubicuos.* Medellín : s.n., 2010. ISSN .
21. DECSAI Departamento de Ciencias de la Computación e I.A. *DECSAI Departamento de Ciencias de la Computación e I.A.* [En línea] <http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf>.
22. **Sommerviller, Ian y Alfonso Galipienso, María Isabel.** *Ingeniería del software.* 2005.
23. **Hurtado Carmona, Dougglas.** *Teoría General de Sistemas: Un Enfoque Hacia La Ingeniería de Sistemas 2ed.* 2011.
24. **DEBRAUWER, Laurent.** *Patrones de diseño para c#.* Barcelona : s.n., 2012. ISBN.
25. **Date, C. J.** *Introducción a los sistemas de bases de datos.* 2001.
26. **Falgueras, Benet Campderrich.** *Ingeniería del software.* 2002.
27. **Alonso Amo, Fernando, Martínez Normand, Loïc y Segovia Pérez, Francisco Javier.** *Introducción a la ingeniería del software.* 2005.
28. [En línea]
29. **Kruchten, Philippe.** The Rational Unified Process: An Introduction. *The Rational Unified Process: An Introduction.* [En línea] 2004. <https://books.google.com.cu/books?id=RYCMx6o47pMC&printsec=frontcover&dq=RUP&hl=es&sa=X&ei=18JIVazKGGKrlsASTw4GACw&ved=0CBsQ6AEwAA#v=onepage&q=RUP&f=false>.

Glosario de Términos

SIG: Sistema de información geográfica.

SIGUCI: Sistema de información geográfica para la Universidad de las Ciencias Informáticas.

LPS: Línea de productos de software.

ESRI: Instituto de Investigación de Sistemas Ambientales.

Atm: Cajeros Automáticos.

ArcGIS: Sistema que permite recopilar, organizar, administrar, analizar, compartir y distribuir información geográfica.